La revue francophone indépendante pour les utilisateurs des Apple][+, //e, //e+, //c™ et Macintosh™

pom's

- Comparateur de disquettes 16 secteurs
- Un nouvel accessoire pour Macintosh
- Le formateur ProDOS désassemblé
- Les ascenseurs en MSBasic
- Courbes fractales en Pascal
- Editeur de blocs ProDOS
- Effets de miroir en HGR
- Hyper-espace et Basic
- Le Disk][sur le //c
- Un micro-finder
- Temps anglais
- Tri de chaînes

NUMERO 25 - PRIX 40 F

M2366-25-40 F

ISSN : 0294-6068

De MacPaint à MacWrite : Pourquoi passer par le bureau ? "Raccourci" évite ce détour.

Le chemin des écoliers, tel est souvent le passage par le Bureau. Pour changer d'application, il suffit aujourd'hui d'appeler l'accessoire "Raccourci". Après avoir confirmé votre intention de quitter le programme en cours, une fenêtre de sélection apparaît et un simple double-clic vous fait passer de Basic à ScreenMaker, de MacPaint à Multiplan et, pourquoi pas, de MacForth à MacPoker...

Des avantages :

Sur un 128 Ko, ne pas passer par le Finder c'est éviter des manipulations gourmandes en temps et en patience.

Sur un 512 Ko, les applications disposent de toute la place mémoire.

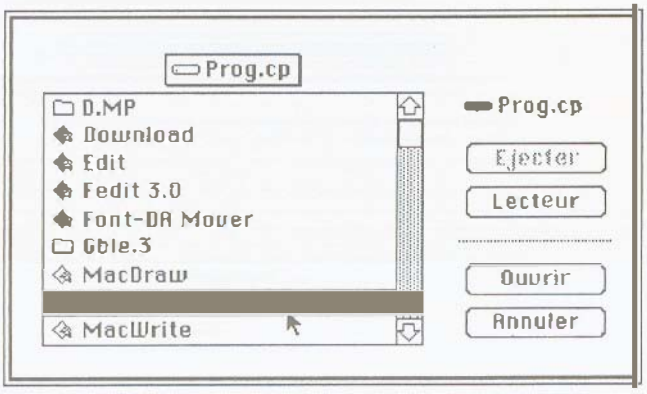
Sur un Mac Plus, les changements sont quasi-instantanés et les 768 Ko de mémoire cache sont disponibles.

Compatible avec Switcher : vous pourrez changer l'une ou l'autre des applications gérées par Switcher directement par cet accessoire.

Compatible disques durs : même si votre disque dur est partagé via AppleTalk...

Disponible à tout moment, "Raccourci" est toujours utilisable car il ne nécessite pas une installation préalable des applications (sait-on toujours à l'avance quelles applications seront nécessaires à un travail donné ?).

Avec Pom's, travaillez *directement* !



Disquette "Raccourci" - 200,00 F franco - Bon de commande page 74 - ●Éditions MEV

Résultats du Sondage — Page 4

Éditorial — Hervé Thiriez — Page 5

Temps Anglais — Yves Wouters — Page 6

Éditeur de blocs ProDOS — Marc Rogliano — Page 9

Janus — Roland Jost — Page 21

Inhibez le 'Reset' en ProDOS — Bruno Fénart — Page 24

CompDisk — Claude Vivier — Page 25

Le formateur ProDOS désassemblé — Bruno Fénart — Page 33

Ad litteram — Julien Thomas — Page 40

Ascenseurs en Basic — Bernard Baz — Page 47

Compatible ? — Hervé Thiriez — Page 49

MacAstuces — Page 50

µFinder — Jean-Luc Bazanegue — Page 51

Tri de chaînes — Bernard Lambillon — Page 52

Un Disk][sur //c — Bruno Fénart — Page 62

Courbes fractales — J.-C. Sorribas — Page 64

Hyper-espace — Olivier Sotiriades — Page 66

Courrier des Lecteurs — Page 68

Micro-informations — Jean-Michel Gourévitch — Page 70

Les annonceurs ; Apple : pages 38 et 39. Infomag : page 76

Éditions MEV - 12, rue d'Anjou - 78000 Versailles. Directeur de la publication : Hervé Thiriez

Diversité

Voici donc ci-contre les résultats chiffrés de notre récent sondage. Les 730 lecteurs qui nous ont répondu ne permettent pas de tirer une valeur représentative (au sens statistique) de ces chiffres ; nous en tirons bien entendu une tendance, confortée par le fait que les résultats sont similaires à ceux de l'an passé.

Toujours aussi peu de femmes – 1,9 % des bulletins – lisent Pom's. Cela correspondrait-il à la proportion de femmes qui s'intéressent à l'informatique ? Notons toutefois que nos 'auteuses' se font aussi prolifiques et assidues que nos auteurs.
Parmi nos lecteurs, 14 % ont un Macintosh, 91,8 % ont un Apple][, 6,03 % ayant les deux appareils. La répartition à l'intérieur de la famille Apple][totalise plus de 100 %, certains ayant deux appareils de ce type, un lecteur sur deux y consacrant entre 3 et 8 heures chaque semaine.
Dans les grandes tendances, il faut aussi remarquer que 58 % des possesseurs d'Apple // ont une carte 80 colonnes étendue et 45 % utilisent régulièrement ProDOS, ce dernier point s'accompagnant de la diminution des][+ au profit des //e.
Un lecteur sur trois possède une souris Apple //, ce qui nous ouvre de nouveaux horizons pour la variété des programmes que nous vous proposerons.
Moins de débutants, plus de lecteurs de niveau intermédiaire et d'amateurs avertis : y sommes-nous pour quelque chose ?

Au-delà de la simple mise à plat des chiffres, ce type d'enquête nous permet, par vos remarques (non quantifiables) de mieux orienter la revue. Bien sûr, la diversité des réponses nous laisse parfois dans une indécision proche de la perplexité mais nous savons rester à l'écoute : un lecteur nous a suggéré de mettre les sources assembleur en format TEXT sur la disquette pour que chacun puisse le charger : chose faite dès le numéro 24.

- de 15 : 2,5	Basic : 98,4	DOS : 66,0	ImageWriter : 54,2
15 - 19 : 5,2	Pascal : 32,6	ProDOS : 45,4	ImageWriter// : 10,4
20 - 24 : 11,5	Forth : 2,2	CP/M : 9,5	Apple DMP : 4,6
25 - 34 : 29,6	Logo : 13,2	80 colonnes : 16,7	Epson : 11,5
35 - 45 : 36,2	A. 6502 : 42,2	80 c. étendue : 58,8	Seikosha : 3,3
+ de 45 : 15,0	68000 : 4,4	c. Z80-CP/M : 25,1	Mannesman : 3,0
masculin : 98,0	Lisp : 0,3	carte souris : 34,6	Oki : 1,4
féminin : 2,0	Cobol : 1,1	disque dur* : 9,8	autre : 11,5
][+ : 17,9	Fortran : 4,4	débutant : 5,2	pub revue : 24,7
//e : 40,6	PLE : 0,0	intermédiaire : 40,0	kiosque : 23,3
//e+ : 17,3	APL : 0,3	amateur averti : 46,0	salon : 2,7
//c : 32,2	Prolog : 0,8	haut niveau : 8,8	boutique : 18,9
/// : 0,6	C : 0,0		amateur : 30,4
Mac128 : 13,6	- de 2 H : 2,4	niveau général : 4,35	intérêt pédag : 3,50
Mac512 : 56,9	3 - 8 H : 51,0	présentation : 3,97	micro-infos : 3,76
MacPlus : 27,5	9 - 16 H : 32,1	intérêt prog. : 3,79	disquettes : 4,12
LISA : 2,0	+ de 16 H : 14,5	clarté articles : 3,43	(Notes entre 0 et 5)

*Disques durs : 9,8 % pour l'ensemble des machines (Mac & Apple //)

Il est vrai que vos compliments nous font rougir (Continuez, meilleure revue, etc.) mais il est également vrai que vos reproches sont indispensables (Pas assez d'explications, soyez plus clairs, il m'a fallu trois jours pour comprendre tel article...) ; nous orientons la barre dans le sens d'une plus grande limpidité par nos 'encadrés-mode d'emploi' – que vous saluez unanimement – et une présentation plus simple.
Sans aucune valeur représentative, mais juste pour le plaisir, voici quelques unes des remarques – arbitrairement choisies – que vous nous avez adressées :

Ce que vous préférez :
«Rêver que j'ai tout compris,
Les applications type 'composeur' (cité au moins une fois sur cinq),
Les utilitaires EPE, MAX,
Les applications pratiques (Multiplan),
Les "ponts" entre DOS et ProDOS,
L'éditorial (!),
Le cahier Mac,
L'absence de publicité,
Les utilitaires à placer dans la pomme du Mac,
Ce qui pallie l'insuffisance des documentations Apple,
Les articles pédagogiques de G. Michel,
La couverture et le prix,
Le sérieux des programmes.»

Ce que vous détestez :
«Les autres (!),
Tous vos assembleurs car j'ai 'The Assembler',
La rédaction érudite d'articles difficilement déchiffrables,
Le prix,
Les anglicismes et le jargon informatique,
Attendre deux mois entre parutions,
Pas assez de publicité,
Ne rien avoir à critiquer, il n'y avait que la présentation mais vous l'avez changée,
La publicité déguisée (NDLR : en quoi ?),
Avoir quatre lignes d'explications pour quatre pages de source,
Ce qui n'est pas programmation (Multiplan).»

Ce qu'il faut développer :
«Intelligence artificielle,
Les astuces,
Un encadré explicatif pour chaque article,
Applications type 'fer à souder',
Applications sur Multiplan, Chart, Draw,
Rien,
Le courrier des lecteurs,
Mettez des organigrammes,
La robotique,
Les logiciels d'application : compte bancaire,
Trucs sur la communication.»

Editorial

Apple Expo à la Villette représente un pas de plus de la part d'Apple vers le professionnalisme : une ambiance beaucoup plus proche de celle du Sicob, et une domination franche et massive du Macintosh. Certes, on parvenait encore à voir des Apple II mais on semblait être dans l'attente du fameux //x, surnommé par certains le Rambo car, tout en restant compatible avec le //e et le //x, il devrait tourner trois fois plus vite.

Par rapport au Sicob d'avril, nous avons vu relativement peu de produits nouveaux. A signaler toutefois pour l'Apple II les excellentes possibilités graphiques du programme Extasie de Créalude – et dont les droits viennent d'être acquis par Apple –, un utilitaire prometteur pour les manipulations en ProDOS, MouseFiler, développé par Froggy Software et, pour les possesseurs de Macintosh, la présentation chez ACI de WriterPlus, dont l'ambition est de remplacer d'un coup tous les MacWrite et autres Word, l'apparition chez Microsoft du Flight Simulator, la version Mac dépassant nettement la version traditionnelle de ce célèbre logiciel de simulation.

Dans ce numéro, nous vous présentons deux nouvelles disquettes : l'une pour l'Apple // est un recueil de trois jeux de réflexion (intense...), œuvre de Sylvie Gallet déjà auteur de programmes pour Pom's. La deuxième, pour le Macintosh, est un accessoire de bureau qui accélère sensiblement les passages d'une application à l'autre : l'art et la manière de passer de MacPaint à MacWrite pour y coller un dessin SANS devoir passer par le bureau électronique. Signé J.-L. Bazanegue.

Ce numéro, certainement influencé par vos nombreuses remarques à l'occasion de notre sondage, est très *ouvert*. Des programmes tout Basic, d'un accès simple, des programmes en assembleur moins évidents à saisir mais d'une utilisation aussi aisée, un programme en Pascal inspiré par les espaces non entiers abordés dans le numéro 22, des ascenseurs Basic, un accessoire de bureau et un micro-Finder pour le Macintosh, et pourquoi pas de l'informatique *hard* de type "fer à souder et pince coupante"...

Avouons-le, certains abonnés ont reçu le précédent numéro de leur revue préférée après mise en kiosque, le routage ayant été trop tardif. *Mea culpa*, on ne nous y reprendra plus !

Hervé Thiriez

Ont collaboré à ce numéro : Alexandre Avrane, Bernard Baz, Jean-Luc Bazanegue, Jean Damiènes, Bruno Fénart, Alain Guillard, Jean-Michel Gourévitch, Olivier Herz, Paule Jade, Roland Jost, Bernard Lambillon, Philippe Meslin, Marc Rogliano, J.-P. Sorribas, Olivier Sotiriades, Julien Thomas, Claude Vivier, Yves Wouters.

Directeur de la publication, rédacteur en chef : Hervé Thiriez.
Rédacteurs : Alexandre Avrane, Olivier Herz.
Siège social : Éditions MEV – 12, rue d'Anjou – 78000 Versailles. Tél : (1) 39 51 24 43.
Publicité : Éditions MEV.
Diffusion : N.M.P.P.
Impression : Rosay – 47, avenue de Paris – 94300 Vincennes. Tél : (1) 43 28 18 63.

Pom's est une revue indépendante, non rattachée à Apple Computer, Inc. ni à Apple Computer France S.A.R.L. Apple, le logo Apple, Mac et le logo Macintosh sont des marques déposées d'Apple Computer, Inc.

Temps anglais

Yves Wouters

Le programme que nous vous proposons ici, s'il ne présente pas de prouesse du point de vue de la programmation, intéressera néanmoins un grand nombre de lecteurs.

Il s'agit d'un jeu éducatif qui affiche un verbe anglais, et vous demande de lui communiquer le preterit, le participe passé et la traduction française. Le programme corrige automatiquement vos réponses incorrectes (ou inexistantes) et affiche votre score à la fin de chaque séquence (de trois à douze questions).

Programme 'TPS PRIMITIFS ANGLAIS'

```
5  REM  (C) 1984 *****
7  REM  WOUTERS Y.****
8  REM  --BELGIQUE--**
10  REM --------TEST ANGLAIS-------
30  DIM VERBE$(196,5)
40  DIM DEJA(196)
50  DIM OK(196)
60  HOME : VTAB 10
70  PRINT "PATIENCE:J'APPRENDS LES T
    EMPS PRIMITIFS."
80  FOR J = 1 TO 191
90  FOR K = 1 TO 5 STEP 1
100  READ VERBE$(J,K)
110  NEXT K
120  NEXT J
210  HOME : VTAB 8
220  PRINT "COMBIEN VOULEZ-VOUS FAIRE
      D'EXERCICES?"
230  PRINT "MINIMUM =3 ET MAXIMUM=12"
     ;
240  INPUT EX
250  IF EX < 3 OR EX > 12 GOTO 230
260 B2 = 0:B3 = 0:B4 = 0
270  FOR J = 1 TO 191:DEJA(J) = 0: NE
     XT J
280  HOME
290  PRINT "INFIN  PRETERIT PARTICIPE
     -TRADUCTION"
300  PRINT "  (TO)            PASSE
      FRANCAISE"
310  PRINT "=========================
     ========"
320  FOR I = 1 TO EX
330  IF TB = 191 THEN  PRINT : PRINT
     "BRAVO!VOUS AVEZ TOUT REUSSI !":
      PRINT "JE N'AI PLUS D'EXERCICES
      A VOUS PROPOSER.": PRINT : PRIN
     T "POU POURSUIVRE,PRESSEZ UNE TO
     UCHE"; GET X$:II = I:I = EX:EX =
      II - 1: GOTO 800
340 J =  INT ( RND (1) * 191) + 1
350  IF DEJA(J) = 1 GOTO 340
360 DEJA(J) = 1
370  IF OK(J) = 3 GOTO 340
380 TE = TE + 1
390  PRINT VERBE$(J,1);
400 PV =  PEEK (37) + 1
410  VTAB 2: HTAB 38: INVERSE : PRINT
      TE;: NORMAL : VTAB PV
420 OK(J) = 0
430  REM -------QUESTION SUR LE PRETE
     RIT.
440  HTAB 8: INPUT "";R2S
450  IF R2$ <  > VERBE$(J,2) GOTO 510
460  REM ----BONNE REPONSE(PASSE SIMP
     LE)
470 B2 = B2 + 1
480 T2 = T2 + 1
490 OK(J) = OK(J) + 1
500  GOTO 550
510  REM ----ERREUR (PASSE SIMPLE)
520  HTAB 8: VTAB PV
530  CALL  - 958
540  INVERSE : PRINT VERBE$(J,2);: NO
     RMAL
550  REM ---QUESTION SUR LE PARTICIPE
      PASSE----
560  HTAB 17: VTAB PV: INPUT "";R3$
570  IF R3$ <  > VERBE$(J,3) GOTO 630
580  REM ---BONNE REPONSE(PARTICIPE P
     ASSE)--
590 B3 = B3 + 1
600 T3 = T3 + 1
610 OK(J) = OK(J) + 1
620  GOTO 670
630  REM ----------ERREUR (PARTICIP
     E PASSE)----
640  HTAB 17: VTAB PV
650  CALL  - 958
660  INVERSE : PRINT VERBE$(J,3);: NO
     RMAL
670  REM ---QUESTION SUR LA TRADUCTIO
     N FRANCAISE---
680  HTAB 27: VTAB PV: INPUT "";R4$
690  IF R4$ <  > VERBE$(J,4) AND R4$
     <  > VERBE$(J,5) GOTO 760
700  REM --------BONNE REPONSE (TRADU
     CTION)--------
710 B4 = B4 + 1
720 T4 = T4 + 1
730 OK(J) = OK(J) + 1
740  IF OK(J) = 3 THEN TB = TB + 1
750  GOTO 800
760  REM ----ERREUR (TRADUCTION)-----
     ----
770  HTAB 27: VTAB PV
780  CALL  - 958
790  INVERSE : PRINT VERBE$(J,4): NOR
     MAL
800  NEXT I
810  PRINT : PRINT : PRINT "ICI :";
820  HTAB 8: PRINT B2;"/";EX;
830  HTAB 17: PRINT B3;"/";EX;
840  HTAB 27: PRINT B4;"/";EX
850  PRINT "TOTAL:";
860  HTAB 8: PRINT T2;"/";TE;
870  HTAB 17: PRINT T3;"/";TE;
880  HTAB 27: PRINT T4;"/";TE
890  PRINT : PRINT "POUR VOIR L'ENSEM
     BLE DE VOS SUCCES,TAPEZ  S ."
900  PRINT "SINON,TAPEZ UN CARACTERE
     QUELCONQUE."
910  GET X$: HOME
920  IF X$ >  < "S" GOTO 970
930  FOR J = 1 TO 191
940  PRINT OK(J);"  "VERBE$(J,1),
950  NEXT J
960  GOTO 890
970  VTAB 23: PRINT "VOULEZ-VOUS
     1.UNE NOUVELLE SERIE
980  PRINT "                  2.ARRETER
     ";
990  GET DECIS$
1000  IF DECIS$ = "1" GOTO 210
1010  IF DECIS$ = "2" THEN  HOME : GOT
      O 1030
1020  HOME : PRINT "REPONDEZ PAR 1 OU
      2 ,S.V.P.": GOTO 970
1030  HOME
1190  END
1195  DATA  ABIDE,ABODE,ABODE,DEMEURER
      ,DEMEURER
1196  DATA  ARISE,AROSE,ARISEN,SE LEVE
      R,SE LEVER
1197  DATA  AWAKE,AWOKE,AWAKEN,S'EVEIL
      LER,S'EVEILLER
1200  DATA  BE,WAS,  BEEN,ETRE,EXISTER
1202  DATA  BEAR,BORE,BORNE,PORTER,POR
      TER
1204  DATA  BEAT,BEAT,BEATEN,BATTRE,BA
      TTRE
1210  DATA  BECOME,BECAME,BECOME,DEVEN
      IR,DEVENIR
1212  DATA  BEFALL,BEFELL,BEFALLEN,ARR
      IVER A,ARRIVER A
1214  DATA  BEGET,BEGOT,BEGOTTEN,ENGEN
      DRER,ENGENDRER
1220  DATA  BEGIN,BEGAN,BEGUN, COMMENC
      ER,COMMENCER
1222  DATA  BEHOLD,BEHELD,BEHELD,CONTE
      MPLER,CONTEMPLER
1224  DATA  BEND,BENT,BENT,COURBER,COU
      RBER
1226  DATA  BEREAVE,BEREFT,BEREFT,PRIV
      ER,PRIVER
1228  DATA  BESEECH,BESOUGHT,BESOUGHT,
      SUPPLIER,SUPPLIER
1229  DATA  BID,BID,BID,ORDONNER,ORDON
```

NER
1230 DATA BIND, BOUND, BOUND, LIER, NOUE R
1232 DATA BITE, BIT, BITTEN, MORDRE, MOR DRE
1234 DATA BLED, BLED, BLED, SAIGNER, SA IGNER
1236 DATA BLOW, BLEW, BLOWN, SOUFFLER, S OUFFLER
1240 DATA BREAK, BROKE, BROKEN, CASSER, ROMPRE
1245 DATA BREED, BRED, BRED, ELEVER, ELE VER
1250 DATA BRING, BROUGHT, BROUGHT, APPO RTER, AMENER
1260 DATA BUILD, BUILT, BUILT, BATIR, CO NSTRUIRE
1262 DATA BURN, BURNT, BURNT, BRULER, BR ULER
1264 DATA BURST, BURST, BURST, ECLATER, ECLATER
1270 DATA BUY, BOUGHT, BOUGHT, ACHETER, ACHETER
1275 DATA CAST, CAST, CAST, JETER, JETER
1280 DATA CATCH, CAUGHT, CAUGHT, ATTRA PER, SAISIR
1285 DATA CHIDE, CHID, CHIDDEN, GRONDER , GRONDER
1290 DATA CHOOSE, CHOSE, CHOSEN, CHOISI R, CHOISIR
1292 DATA CLEAVE, CLEFT, CLEFT, FENDRE , FENDRE
1294 DATA CLING, CLUNG, CLUNG, S'ACCROC HER, S'ACCROCHER
1296 DATA CLOTHE, CLED, CLAD, VETIR, VET IR
1300 DATA COME, CAME, COME, VENIR, VENIR
1310 DATA COST, COST, COST, COUTER, COUT ER
1312 DATA CREEP, CREPT, CREPT, RAMPER, R AMPER
1314 DATA CROW, CREW, CROWED, CHANTER, C HANTER
1320 DATA CUT, CUT, CUT, COUPER, COUPER
1322 DATA DARE, DURST, DARED, OSER, OSER
1324 DATA DEAL, DEALT, DEALT, DISTRIBUE R, DISTRIBUER
1326 DATA DIG, DUG, DUG, CREUSER, CREUS ER
1330 DATA DO, DID, DONE, FAIRE, FAIRE
1340 DATA DRAW, DREW, DRAWN, DESSINER, T IRER
1350 DATA DREAM, DREAMT, DREAMT, REVER, REVER
1360 DATA DRINK, DRANK, DRUNK, BOIRE, BO IRE
1370 DATA DRIVE, DROVE, DRIVEN, POUSSER , CONDUIRE
1375 DATA DWELL, DWELT, DWELT, DEMEURER , DEMEURER
1380 DATA EAT, ATE, EATEN, MANGER, MANGE R
1390 DATA FALL, FELL, FALLEN, TOMBER, TO MBER
1400 DATA FEED, FED, FED, NOURRIR, ALIME NTER
1410 DATA FEEL, FELT, FELT, SENTIR, RESS ENTIR
1420 DATA FIGHT, FOUGHT, FOUGHT, COMBAT TRE, SE BATTRE
1430 DATA FIND, FOUND, FOUND, TROUVER, T ROUVER
1432 DATA FLEE, FLED, FLED, FUIR, FUIR
1434 DATA FLING, FLUNG, FLUNG, LANCER, L ANCER
1440 DATA FLY, FLEW, FLOWN, VOLER, VOLER
1442 DATA FORBEAR, FORBORE, FORBORNE, S 'ABSTENIR, S'ABSTENIR
1444 DATA FORBID, FORBADE, FORBIDDEN, D EFENDRE, DEFENDRE
1450 DATA FORGET, FORGOT, FORGOTTEN, OU BLIER, OUBLIER
1452 DATA FORGIVE, FORGAVE, FORGIVEN, P ARDONNER, PARDONNER
1454 DATA FORSAKE, FORSOOK, FORSAKEN, A BANDONNER, ABANDONNER
1460 DATA FREEZE, FROZE, FROZEN, GELER, CONGELER
1470 DATA GET, GOT, GOT, OBTENIR, DEVENI R
1472 DATA GILD, GILT, GILT, DORER, DORER
1474 DATA GIRD, GIRT, GIRT, CEINDRE, CEI NDRE
1480 DATA GIVE, GAVE, GIVEN, DONNER, DON NER
1490 DATA GO, WENT, GONE, ALLER, ALLER
1495 DATA GRIND, GROUND, GROUND, MOUDRE , MOUDRE
1500 DATA GROW, GREW, GROWN, CROITRE, GR ANDIR
1510 DATA HANG, HUNG, HUNG, PENDRE, PEND RE
1520 DATA HAVE, HAD, HAD, AVOIR, AVOIR
1530 DATA HEAR, HEARD, HEARD, ENTENDRE, OUIR
1532 DATA HEAVE, HOVE, HOVE, SE SOULEVE R, SE SOULEVER
1534 DATA HEW, HEWED, HEWN, TAILLER, TAI LLER
1540 DATA HIDE, HID, HIDDEN, CACHER, CAC HER
1550 DATA HIT, HIT, HIT, FRAPPER, FRAPPE R
1560 DATA HOLD, HELD, HELD, TENIR, MAINT ENIR
1570 DATA HURT, HURT, HURT, BLESSER, FAI RE SOUFFRIR
1580 DATA KEEP, KEPT, KEPT, GARDER, GARD ER
1582 DATA KNEEL, KNELT, KNELT, S'AGENOU ILLER, S'AGENOUILLER
1584 DATA KNIT, KNIT, KNIT, TRICOTER, TR ICOTER
1590 DATA KNOW, KNEW, KNOWN, CONNAITRE, SAVOIR
1592 DATA LADE, LADED, LADEN, CHARGER, C HARGER
1594 DATA LAY, LAID, LAID, POSER, POSER
1600 DATA LEAD, LED, LED, CONDUIRE, DIRI GER
1602 DATA LEAN, LEANT, LEANT, SE PENCHE R, SE PENCHER
1604 DATA LEAP, LEAPT, LEAPT, BONDIR, BO NDIR
1610 DATA LEARN, LEARNT, LEARNT, APPREN DRE, APPRENDRE
1620 DATA LEAVE, LEFT, LEFT, LAISSER, QU ITTER
1630 DATA LEND, LENT, LENT, PRETER, PRET ER
1640 DATA LET, LET, LET, LAISSER, PERMET TRE
1642 DATA LIE, LAY, LAIN, ETRE COUCHE, E TRE COUCHER
1644 DATA LIGHT, LIT, LIT, ALLUMER, ALUM ER
1650 DATA LOSE, LOST, LOST, PERDRE, PERD RE
1660 DATA MAKE, MADE, MADE, FAIRE, FABRI QUER
1670 DATA MEAN, MEANT, MEANT, SIGNIFIER , VOULOIR DIRE
1680 DATA MEET, MET, MET, RENCONTRER, RE NCONTRER
1682 DATA MISTAKE, MISTOOK, MISTAKEN, S E TROMPER, SE TROMPER
1684 DATA MOW, MOWED, MOWN, FAUCHER, FAU CHER
1690 DATA PAY, PAID, PAID, PAYER, PAYER
1695 DATA PEN, PENT, PENNED, PARQUER, PA RQUER
1700 DATA PUT, PUT, PUT, METTRE, PLACER
1710 DATA READ, READ, READ, LIRE, LIRE
1712 DATA REND, RENT, RENT, DECHIRER, DE CHIRER
1714 DATA RID, RID, RID, DEBARRASSER, DE BARRASSER
1720 DATA RIDE, RODE, RIDDEN, ALLER A C HEVAL, ALLER A VELO
1730 DATA RING, RANG, RUNG, SONNER, SONN ER
1740 DATA RISE, ROSE, RISEN, SE LEVER, S E LEVER
1750 DATA RUN, RAN, RUN, COURIR, COURIR
1755 DATA SAW, SAWED, SAWN, SCIER, SCIER
1760 DATA SAY, SAID, SAID, DIRE, DIRE
1770 DATA SEE, SAWN, SEEN, VOIR, VOIR
1772 DATA SEEK, SOUGHT, SOUGHT, CHERCHE R, CHERCHER
1774 DATA SEETHE, SOD, SODDEN, BOUILLIR , BOUILLIR
1780 DATA SELL, SOLD, SOLD, VENDRE, VEND RE
1790 DATA SEND, SENT, SENT, ENVOYER, ENV OYER
1800 DATA SET, SET, SET, POSER, PLACER
1810 DATA SHAKE, SHOOKE, SHAKEN, SECOUE R, SECOUER
1812 DATA SHEAR, SHORE, SHORN, TONDRE, T ONDRE
1814 DATA SHED, SHED, SHED, VERSER, VERS ER
1820 DATA SHINE, SHONE, SHONE, BRILLER, LUIRE
1825 DATA SHOE, SHOD, SHOD, CHAUSSER, CH AUSSER
1830 DATA SHOOT, SHOT, SHOT, TIRER, DONN ER UN COUP DE PIED
1840 DATA SHOW, SHOWED, SHOWN, MONTRER, EXPOSER
1842 DATA SHRED, SHRED, SHRED, MONTRER, MONTER
1844 DATA SHRINK, SHRANK, SHRUNK, SE RE TRECIR, SE RETRECIR
1846 DATA SHRIVE, SHROVE, SHRIVEN, CONF ESSER, CONFESSER
1850 DATA SHUT, SHUT, SHUT, FERMER, CLOR E
1860 DATA SING, SANG, SUNG, CHANTER, CHA NTER
1865 DATA SINK, SANK, SUNK, ENFONCER, EN FONCER
1870 DATA SIT, SAT, SAT, ETRE ASSIS, ET RE ASSIS
1875 DATA SLAY, SLEW, SLAIN, TUER, TUER
1880 DATA SLEEP, SLEPT, SLEPT, DORMIR, D ORMIR
1881 DATA SLIDE, SLID, SLID, GLISSER, GL ISSER
1882 DATA SLING, SLUNG, SLUNG, LANCER, L ANCER
1883 DATA SLINK, SLUNK, SLUNK, SE GLISS ER, SE GLISSER
1884 DATA SLIT, SLIT, SLIT, FENDRE, FEND RE
1885 DATA SMELL, SMELT, SMELT, SENTIR, S ENTIR
1886 DATA SMITE, SMOTE, SMITTEN, FRAPPE R, FRAPPER
1887 DATA SOW, SOWED, SOWN, SEMER, SEMER
1890 DATA SPEAK, SPOKE, SPOKEN, PARLER, PARLER
1895 DATA SPEED, SPED, SPED, SE HATER, S E HATER

```
1900 DATA SPELL,SPELT,SPELT,EPELER,O
     RTHOGRAPHIER
1910 DATA SPEND,SPENT,SPENT,DEPENSER
     ,DEPENSER
1912 DATA SPILL,SPILT,SPILT,REPANDRE
     ,REPANDRE
1914 DATA SPIN,SPUN,SPUN,FILER,TOURN
     ER
1916 DATA SPIT,SPIT,SPIT,CRACHER,CRA
     CHER
1918 DATA SPLIT,SPLIT,SPLIT,FENDRE,F
     ENDRE
1920 DATA SPOIL,SPOILT,SPOILT,GATER,
     ABIMER
1922 DATA SPREAD,SPREAD,SPREAD,ETEND
     RE,ETENDRE
1924 DATA SPRING,SPRANG,SPRUNG,S'ELA
     NCER,S'ELANCER
1930 DATA STAND,STOOD,STOOD,SE TENIR
      DEBOUT,SE DRESSER
1932 DATA STAY,STAID,STAID,SÈJOURNER
     ,RESTER
1934 DATA STEAL,STOLÈ,STOLEN,VOLER,D
     EROBER
1940 DATA STICK,STUCK,STUCK,ADHERER,
     COLLER
1942 DATA STING,STUNG,STUNG,PIQUER,P
     IQUER
1944 DATA STINK,STANK,STUNK,PUER,PUE
     R
1946 DATA STREW,STREWED,STREWN,JONCH
     ER,JONCHER
1948 DATA STRIDE,STRODE,STRIDDEN,MAR
     CHER A GRANDS PAS,SE DEPECHER
1950 DATA STRIKE,STRUCK,STRUCK,FRAPP
     ER,FAIRE GREVE
1951 DATA STRING,STRUNG,STRUNG,ENFIL
     ER,ENFILER
1952 DATA STRIVE,STROVE,STRIVEN,S'EF
     FORCER,S'EFFORCER
1953 DATA SWEAR,SWORE,SWORN,JURER,JU
     RER
1954 DATA SWEAT,SWEAT,SWEAT,SUER,SUE
     R
1955 DATA SWEEP,SWEPT,SWEPT,BALAYER,
     BALAYER
1956 DATA SWELL,SWELLED,SWOLLEN,ENFL
     ER,ENFLER
1960 DATA SWIM,SWAM,SWUM,NAGER,NAGER
1965 DATA SWING,SWUNG,SWUNG,BALANCER
     ,BALANCER
1970 DATA TAKE,TOOK,TAKEN,PRENDRE,PR
     ENDRE
1980 DATA TEACH,TAUGHT,TAUGHT,ENSEIG
     NER,ENSEIGNER
1990 DATA TEAR,TORE,TORN,DECHIRER,D
     ECHIRER
2000 DATA TELL,TOLD,TOLD,DIRE,DIRE
2010 DATA THINK,THOUGHT,THOUGHT,PENS
     ER,REFLECHIR
2015 DATA THRIVE,THROVE,THRIVEN,PROS
     PERER,PROSPERER
2020 DATA THROW,THREW,THROWN,JETER,L
     ANCER
2022 DATA THRUST,THRUST,THRUST,LANCE
     R,LANCER
2024 DATA TREAD,TROD,TROD,FOULER,FOU
     LER AUX PIEDS
2026 DATA UNDERSTAND,UNDERSTOOD,UNDE
     RSTOOD,COMPRENDRE,COMPRENDRE
2028 DATA UNDO,UNDID,UNDONE,DEFAIRE,
     DEFAIRE
2029 DATA UPSET,UPSET,UPSET,RENVERSE
     R,RENVERSER
2030 DATA WAKE,WOKE,WOKEN,EVEILLER,S
     'EVEILLER
2040 DATA WEAR,WORE,WORN,PORTER,PORT
     ER UN VETEMENT
2042 DATA WEAVE,WOVE,WOVEN,TISSER,TI
     SSER
2044 DATA WEEP,WEPT,WEPT,PLEURER,PLE
     URER
2050 DATA WIN,WON,WON,GAGNER,GAGNER
2052 DATA WIND,WOUND,WOUND,ENROULER,
     ENROULER
2054 DATA WITHDRAW,WITHDREW,WITHDRAW
     ,RETIRER,RETIRER
2056 DATA WITHSTAND,WITHSTOOD,WITHST
     OOD,RESISTER A,RESISTER A
2058 DATA WORK,WROUGHT,WROUGHT,TRAVA
     ILLER,TRAVAILLER
2059 DATA WRING,WRUNG,WRUNG,TORDRE,T
     ORDRE
2060 DATA WRITE,WROTE,WRITTEN,ECRIRE
     ,ECRIRE
2062 DATA WRITHE,WRITHED,WRITHEN,SE
     TORDRE,SE TORDRE
2400 '
2500 '(C)1984 -WOUTERS YVES
```

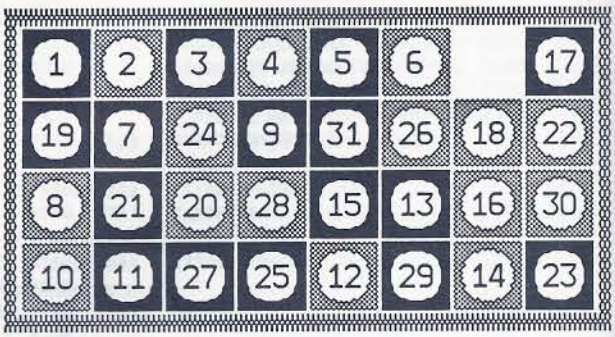
Une nouvelle disquette de jeux : Ludologic

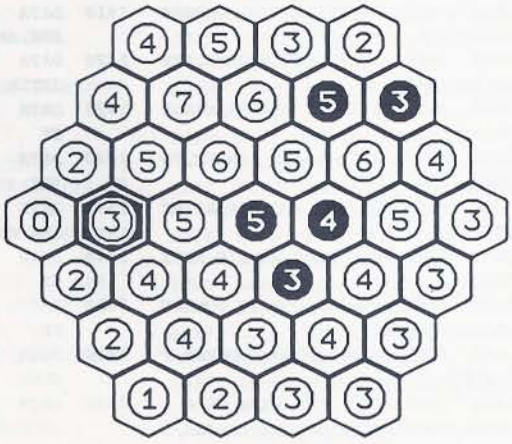
Au sommaire de la disquette LUDOLOGIC, trois jeux de réflexion de difficulté croissante. Ces jeux qui nécessitent des neurones aussi calmes qu'entrainés, ne devraient pas décevoir les amateurs de puzzles et autres casse-têtes.

Il n'est pas nécessaire de présenter TAQUIN, ce pousse-pousse informatique ici fort bien présenté.

Nouvelle difficulté, NOIR & BLANC : 37 hexagones peuvent être noirs ou blancs mais au départ vous n'en connaissez pas la couleur. Chacun comporte un numéro qui représente le nombre de cellules voisines blanches... A vous de reconstituer le décor original !

HEXAGONE MAGIQUE est encore plus délicat, même principe que le carré magique, mais ici vous devrez installer les chiffres de 1 à 19 dans un hexagone de telle façon que les 5 horizontales et 10 obliques totalisent chacune 38 : bonne chance.





80,00 F Franco,
Bon de commande page 74

Fidèle à son habitude, Pom's vous propose sur cette disquette les sources des routines écrites par Sylvie Gallet en assembleur Lisa 2.5. Bien entendu, le Basic est également listable. TAQUIN et NOIR & BLANC utilisent leur propre routine graphique qui permet de dessiner plus rapidement qu'avec des shapes.

Editeur de blocs ProDOS

Marc Rogliano

Ce programme, écrit en assembleur, permet la lecture et l'écriture directe sur l'un des 280 blocs d'une disquette ProDOS. (RSPD : Read and Save ProDos).

Utilisation

L'affichage 80 colonnes est indispensable. Sur un Apple //c ou un Apple //e mis à niveau il utilise les caractères souris ; toutefois il fonctionne sans problème sur un ancien //e, la présentation étant simplement moins agréable.
Il faut sauvegarder le fichier objet (R.S.P.D.OBJET) sur une disquette ProDOS à l'aide des 'Utilitaires Système' avant de lancer le programme.
Ceci fait, exécutez le programme. Le message de présentation doit alors apparaître. Insérez ensuite la disquette ProDOS que vous voulez lire ou modifier dans le lecteur 1 (lecteur integré sur un //c) et appuyez sur <RETURN>. La disquette se met à tourner et le contenu du block 0 apparaît sur l'écran.

Commandes

- **flèches** : déplacement du curseur sur l'écran ;
- ⌘ et flèches gauche ou droite : lecture du bloc précédant ou du bloc suivant ;
- ⇧ et flèches gauche ou droite : lecture des blocs précédant ou suivant par pas de 10 ;
- **<espace>** : changement de page pour lire le premier ou le second groupe de 256 octets (1 bloc contient 512 octets) ;
- Lorsque le curseur est sur un octet, on peut modifier ce dernier en tapant directement le code hexadécimal voulu ;
- **CTRL-A** : entrée dans le mode d'insertion des caractères (jusqu'à un nouveau CTRL-A) ;
- **CTRL-B** : entrée du numéro de bloc pour que son contenu soit affiché sur l'écran (et ceci sans passer par les touches ⌘ et ⇧).
- **CTRL-C** : changement de lecteur (Drive 1-2).
- **CTRL-S** : sauvegarde des modifications éventuelles sur la disquette (attention : l'ancien contenu du bloc est alors perdu).
- **CTRL-X** : switch entre deux types de caractères :
  - Tous les caractères pouvant être affichés le sont (y compris les caractères souris).
  - Les caractères affichés sont obtenus par 'ORA #$80' pour mettre leur bit 7 à un (les valeurs '00' sont alors affichées sous la forme du caractère ".").
- **ESC**, enfin, pour quitter le programme.

Précisions

Il convient de souligner ici quelques points :

- d'abord, un emprunt : la routine sonore du 'Moniteur étendu' de Thierry le Tallec (Pom's 8) ;
- le buffer utilisé va de l'adresse $4000 à l'adresse $41FF (un bloc ProDOS équivaut à 2 secteur DOS 3.3 c'est-à-dire à $200 octets) ;
- au démarrage, le programme teste s'il y a une carte 80 colonnes et s'arrête dans le cas contraire ; d'autre part, il teste également la machine sur laquelle il tourne et s'il peut utiliser les icônes souris, dans le cas contraire, ceux-ci seront remplacés par des caractères standards comme, par exemple, '-' ou '!' pour les cadres ;
- enfin, les erreurs concernant le disque sont détectées et affichées.

Note : on peut aussi lire des disques formattés en DOS 3.3 ou PASCAL avec R.S.P.D.OBJET, mais il faut alors se rappeler la correspondance entre les blocs ProDOS et les secteurs logiques des autres systèmes d'exploitation (voir Pom's n° 18 et 19).

Récapitulation R.S.P.D.OBJET

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE R.S.P.D.OBJET,A$6000,L$F45

```
6000- 20 39 FB A9 FF 85 32 20
6008- 58 FC AD 98 BF 29 02 D0
6010- 17 A9 19 A0 6F A2 07 86
6018- 06 86 07 20 D4 69 20 B9
6020- 69 20 B9 69 20 B9 69 60
6028- 38 AD C0 FB F0 05 C9 E0
6030- F0 01 18 66 09 A2 03 B5
6038- 36 9D 3A 6F CA 10 F8 A9
6040- 03 20 95 FE A9 11 20 ED
6048- FD 20 58 FC A9 01 85 06
6050- A9 01 85 07 A9 2A A0 6A
6058- 20 D4 69 A9 91 A0 6A 20
6060- D4 69 A9 F5 A0 6A 20 D4
6068- 69 A9 7A A0 6B 20 D4 69
6070- A9 FB A0 6B 24 09 30 04
6078- A9 1D A0 6C 20 D4 69 20
6080- B9 69 8D 10 C0 AD 00 C0
6088- 10 FB C9 8D F0 09 8D 10
6090- C0 20 B9 69 4C 85 60 20
6098- 58 FC A9 12 20 ED FD 20
60A0- 58 FC A9 04 85 1E A9 03
60A8- 8D 3E 6F A9 60 8D 3F 6F
60B0- A9 00 8D 40 6F 8D 42 6F
60B8- 8D 43 6F A9 40 8D 41 6F
60C0- 20 00 BF 80 3E 6F 90 06
```

Source 'R.S.P.D.SOURCE'

Assembleur ProCODE

```
        DSK  R.S.P.D.OBJET
****************************
**........................**
**..UTILITAIRE..R.S.P.D...**
**..(READ.&.SAVE.PRODOS)..**
**........MARC.ROGLIANO...**
**........................**
****************************
*.Asm.Procode
        ORG  $6000
************************
*..ADRESSES.PAGE.ZERO..*
************************
COL      EQU  $06
LIG      EQU  $07
CAR      EQU  $08
ICONE    EQU  $09
AFFI.L   EQU  $1A
AFFI.H   EQU  $1B
SWITCH   EQU  $1C
CURS     EQU  $1E
LB       EQU  $EB
HB       EQU  $EC
LB.      EQU  $ED
HB.      EQU  $EE
INTER    EQU  $EF
DISK.L   EQU  $FA
DISK.H   EQU  $FB
CURS.COL EQU  $FC
CURS.LIG EQU  $FD
L.LB     EQU  $FE
L.HB     EQU  $FF
**********************
*..ADRESSES.SYSTEME..*
**********************
STO80.1  EQU  $C001
STO80.0  EQU  $C000
PAGE1    EQU  $C054
PAGE2    EQU  $C055
BUTTON0  EQU  $C061
BUTTON1  EQU  $C062
*************
*..ROUTINES..*
*************
HOME     EQU  $FC58
SETTXT   EQU  $FB39
COUT     EQU  $FDED
SPKR     EQU  $C030
BASCALC  EQU  $FBC1
PRBYTE   EQU  $FDDA
FRMNUM   EQU  $DD67
AYINT    EQU  $E10C
LINPTR   EQU  $ED24
GIVAYF   EQU  $E2F2
MOVAF    EQU  $EB63
FDIVT    EQU  $EA69
FADDT    EQU  $E7C1
OUTPRT   EQU  $FE95
*******************
*..INITIALISATION..*
*.//c.ou.new.//e.?.*
*******************
         JSR  SETTXT      ;TEXT
         LDA  #$FF
         STA  $32         ;NORMAL
         JSR  HOME
         LDA  $BF98
         AND  #%00000010
         BNE  OLISGOUD
         LDA  #<MESSERR
         LDY  #>MESSERR
         LDX  #$07
         STX  COL
         STX  LIG
         JSR  STROUT40
         JSR  SOUND
         JSR  SOUND
         JSR  SOUND
         RTS
OLISGOUD SEC
         LDA  $FBC0
         BEQ  IIC
         CMP  #$E0
         BEQ  NEW.IIE
         CLC
NEW.IIE  EQU  *
IIC      ROR  ICONE
         LDX  #3
SAVEIO   LDA  $36,X
         STA  SAVE,X
         DEX
         BPL  SAVEIO
         LDA  #3
         JSR  OUTPRT
*******************************
*.AFFICHAGE.DE.LA.PRESENTATION.*
*******************************
DEB.80   LDA  #17
         JSR  COUT
         JSR  HOME
         LDA  #$01
         STA  COL
         LDA  #$01
         STA  LIG
         LDA  #<PRES.0
         LDY  #>PRES.0
         JSR  STROUT40
         LDA  #<PRES.1
         LDY  #>PRES.1
         JSR  STROUT40
         LDA  #<PRES.2
         LDY  #>PRES.2
         JSR  STROUT40
         LDA  #<PRES.3
         LDY  #>PRES.3
         JSR  STROUT40
         LDA  #<PRES.4.1
         LDY  #>PRES.4.1
         BIT  ICONE
         BMI  PRESICON
         LDA  #<PRES.4.2
         LDY  #>PRES.4.2
PRESICON JSR  STROUT40
         JSR  SOUND
         STA  $C010
W.KBD    LDA  $C000
         BPL  W.KBD
         CMP  #$8D
         BEQ  XXX
         STA  $C010
         JSR  SOUND
         JMP  W.KBD
************************
*.CHARGEMENT.DU.BLOCK.0.*
************************
XXX      JSR  HOME
         LDA  #18
         JSR  COUT
         JSR  HOME
```

```
60C8- 20 BF 68 4C 44 60 A9 00
60D0- 85 06 A9 01 85 07 A9 49
60D8- A0 6C 20 00 6A A9 A9 A0
60E0- 6C 24 09 30 04 A9 B1 A0
60E8- 6C 20 00 6A A9 B9 A0 6C
60F0- 20 00 6A A9 01 A0 6D 24
60F8- 09 30 04 A9 25 A0 6D 20
6100- 00 6A A9 49 A0 6D 20 00
6108- 6A A9 78 A0 6D 24 09 30
6110- 04 A9 9C A0 6D 20 00 6A
6118- A9 C0 A0 6D 20 00 6A 20
6120- DC 65 A9 01 85 1C A9 40
6128- 85 FF A9 00 85 FE 85 FC
6130- 85 FD 85 FB 85 FA 20 5F
6138- 67 20 32 67 20 B9 69 8D
6140- 10 C0 AD 00 C0 10 FB C9
6148- 88 D0 03 4C C9 61 C9 95
6150- D0 03 4C F1 61 C9 8B D0
6158- 03 4C 1B 62 C9 8A D0 03
6160- 4C 5C 62 C9 A0 D0 03 4C
6168- 38 63 C9 81 D0 03 4C 74
6170- 63 C9 82 D0 03 4C 13 64
6178- C9 83 D0 03 4C B8 64 C9
6180- 93 D0 03 4C 06 65 C9 98
6188- D0 03 4C 20 65 C9 9B D0
6190- 03 4C 38 65 20 A5 61 90
6198- 03 4C 5A 65 20 B9 69 8D
61A0- 10 C0 4C 42 61 C9 B0 90
61A8- 0C C9 C7 B0 08 C9 C1 B0
61B0- 06 C9 BA 90 02 18 60 38
61B8- 60 C9 8D F0 08 C9 B0 90
61C0- 06 C9 BA B0 02 18 60 38
61C8- 60 AD 61 C0 10 03 4C E6
61D0- 62 AD 62 C0 10 03 4C 96
61D8- 62 A5 FC D0 09 20 B9 69
61E0- 8D 10 C0 4C 42 61 C6 FC
61E8- 20 DF 67 8D 10 C0 4C 42
61F0- 61 AD 61 C0 10 03 4C 0F
61F8- 63 AD 62 C0 10 03 4C BD
6200- 62 A5 FC C9 0F 90 09 20
6208- B9 69 8D 10 C0 4C 42 61
6210- E6 FC 20 DF 67 8D 10 C0
6218- 4C 42 61 A5 FD D0 09 20
6220- B9 69 8D 10 C0 4C 42 61
6228- 0A 0A 0A DA B0 0C C5 FE
6230- D0 1F 38 E9 10 85 FE 4C
6238- 46 62 C5 FE D0 13 A9 40
6240- 85 FF A9 F0 85 FE C6 FD
```

```
        LDA  #$04
        STA  CURS
        LDA  #$03
        STA  TABLE
        LDA  #$60
        STA  TABLE+1
        LDA  #$00
        STA  TABLE+2
        STA  TABLE+4
        STA  TABLE+5
        LDA  #$40
        STA  TABLE+3
        JSR  $BF00
        DFB  $80
        OA   TABLE
        BCC  AFF.MENU
        JSR  ERROR
        JMP  DEB.80
****************************
*.AFICHAGE.COMMANDES+CADRE.*
****************************
AFF.MENU LDA  #$00
        STA  COL
        LDA  #$01
        STA  LIG
        LOA  #<AAA0
        LDY  #>AAA0
        JSR  STROUT80
        LDA  #<AAA0.1
        LDY  #>AAA0.1
        BIT  ICONE
        BMI  AAAI0
        LDA  #<AAA0.2
        LDY  #>AAA0.2
AAAI0   JSR  STROUT80
        LDA  #<AAA1
        LDY  #>AAA1
        JSR  STROUT80
        LOA  #<AAA1.1
        LDY  #>AAA1.1
        BIT  ICONE
        BMI  AAAI1
        LDA  #<AAA1.2
        LDY  #>AAA1.2
AAAI1   JSR  STROUT80
        LDA  #<AAA2
        LDY  #>AAA2
        JSR  STROUT80
        LOA  #<AAA2.1
        LDY  #>AAA2.1
        BIT  ICONE
        BMI  AAAI2
        LDA  #<AAA2.2
        LDY  #>AAA2.2
AAAI2   JSR  STROUT80
        LDA  #<AAA3
        LDY  #>AAA3
        JSR  STROUT80
        JSR  CADRE
        LDA  #$01
        STA  SWITCH
        LDA  #$40
        STA  L.HB
        LDA  #$00
        STA  L.LB
        STA  CURS.COL
        STA  CURS.LIG
        STA  DISK.H
        STA  DISK.L
        JSR  AFF...
        JSR  AFF.
        JSR  SOUND
        STA  $C010
*************************
*.ATTENTE.D'UNE.COMMANDE.*
*************************
WAIT.KBD LDA  $C000
        BPL  WAIT.KBD
        CMP  #$88
        BNE  XXX.0
        JMP  GAUCHE
XXX.0   CMP  #$95
        BNE  XXX.1
        JMP  DROITE
XXX.1   CMP  #$8B
        BNE  XXX.2
        JMP  HAUT
XXX.2   CMP  #$8A
        BNE  XXX.3
        JMP  BAS
XXX.3   CMP  #$A0
        BNE  XXX.4
        JMP  SPACE
XXX.4   CMP  #$81
        BNE  XXX.5
        JMP  C.A.
XXX.5   CMP  #$82
        BNE  XXX.6
        JMP  C.B.
XXX.6   CMP  #$83
        BNE  XXX.7
        JMP  C.C.
XXX.7   CMP  #$93
        BNE  XXX.8
        JMP  C.S.
XXX.8   CMP  #$98
        BNE  XXX.9
        JMP  C.X.
XXX.9   CMP  #$9B
        BNE  XXX.10
        JMP  ESC
XXX.10  JSR  TEST
        BCC  XXX.11
        JMP  ENT.CAR
XXX.11  JSR  SOUND
        STA  $C010
        JMP  WAIT.KBD
*******************************
*.CARACTERE.ENTRE.0-9.OU.A-F.?.*
*******************************
TEST    CMP  #$B0
        BCC  TEST0
        CMP  #$C7
        BCS  TEST0
        CMP  #$C1
        BCS  TEST1
        CMP  #$BA
        BCC  TEST1
TEST0   CLC
        RTS
TEST1   SEC
        RTS
****************************
*.CARACTERE.ENTRE.0.ET.9.?.*
****************************
TEST..  CMP  #$8D
        BEQ  TEST..1
        CMP  #$B0
        BCC  TEST..2
        CMP  #"9"+1
        BCS  TEST..2
TEST..1 CLC
        RTS
TEST..2 SEC
        RTS
*******************************
*DEPLACEMENTS.CURSEUR.SUR.ECRAN*
```

```
6248- 20 32 67 8D 10 C0 4C 42
6250- 61 C6 FD 20 DF 67 8D 10
6258- C0 4C 42 61 A5 FD C9 1F
6260- 90 09 20 B9 69 8D 10 C0
6268- 4C 42 61 E6 FD A5 FD 0A
6270- 0A 0A 0A 90 18 C5 FE D0
6278- 14 18 69 10 85 FE D0 04
6280- A9 41 85 FF 20 32 67 8D
6288- 10 C0 4C 42 61 20 DF 67
6290- 8D 10 C0 4C 42 61 A5 FB
6298- D0 0D A5 FA D0 09 20 B9
62A0- 69 8D 10 C0 4C 42 61 38
62A8- A5 FA E9 01 85 FA A5 FB
62B0- E9 00 85 FB 20 39 67 8D
62B8- 10 C0 4C 42 61 A5 FB F0
62C0- 0F A5 FA C9 17 90 09 20
62C8- B9 69 8D 10 C0 4C 42 61
62D0- 18 A5 FA 69 01 85 FA A5
62D8- FB 69 00 85 FB 20 39 67
62E0- 8D 10 C0 4C 42 61 A5 FB
62E8- D0 0F A5 FA C9 0A B0 09
62F0- 20 B9 69 8D 10 C0 4C 42
62F8- 61 38 A5 FA E9 0A 85 FA
6300- A5 FB E9 00 85 FB 20 39
6308- 67 8D 10 C0 4C 42 61 A5
6310- FB F0 0F A5 FA C9 0E 90
6318- 09 20 B9 69 8D 10 C0 4C
6320- 42 61 18 A5 FA 69 0A 85
6328- FA A5 FB 69 00 85 FB 20
6330- 39 67 8D 10 C0 4C 42 61
6338- A5 FF C9 41 B0 1E A9 41
6340- 85 FF A9 00 85 FE A5 FD
6348- C9 10 B0 07 18 A9 10 65
6350- FD 85 FD 20 32 67 8D 10
6358- C0 4C 42 61 A9 40 85 FF
6360- A9 00 85 FE 38 A5 FD E9
6368- 10 85 FD 20 32 67 8D 10
6370- C0 4C 42 61 8D 10 C0 20
6378- 28 68 A5 FC A2 03 CA F0
6380- 06 18 65 FC 4C 7E 63 18
6388- 69 09 85 06 A5 FD 0A 0A
6390- 0A DA 38 E5 FE 4A 4A 4A
6398- 4A 18 69 08 85 07 A9 3F
63A0- 85 08 20 3C 69 E6 06 20
63A8- 3C 69 AD 00 C0 10 FB 8D
63B0- 10 C0 C9 81 F0 54 85 EF
63B8- A5 FD 0A 0A 0A 0A 18 65
63C0- FC 38 E5 FE A8 A5 EF 91
```

```
*******************************
GAUCHE   LDA  BUTTON0
         BPL  XXX.A
         JMP  GAUCHE.
XXX.A    LDA  BUTTON1
         BPL  XXX.B
         JMP  GAUCHE.
XXX.B    LDA  CURS.COL
         BNE  GAUCHE1
         JSR  SOUND
         STA  $C010
         JMP  WAIT.KBD
GAUCHE1  DEC  CURS.COL
         JSR  AFF....
         STA  $C010
         JMP  WAIT.KBD
DROITE   LDA  BUTTON0
         BPL  XXX.C
         JMP  DROITE..
XXX.C    LDA  BUTTON1
         BPL  XXX.D
         JMP  DROITE.
XXX.D    LDA  CURS.COL
         CMP  #$0F
         BCC  DROITE1
         JSR  SOUND
         STA  $C010
         JMP  WAIT.KBD
DROITE1  INC  CURS.COL
         JSR  AFF....
         STA  $C010
         JMP  WAIT.KBD
HAUT     LDA  CURS.LIG
         BNE  HAUT1
         JSR  SOUND
         STA  $C010
         JMP  WAIT.KBD
HAUT1    ASL
         ASL
         ASL
         ASL
         BCS  HAUT2
         CMP  L.LB
         BNE  HAUT4
         SEC
         SBC  #$10
         STA  L.LB
         JMP  HAUT3
HAUT2    CMP  L.LB
         BNE  HAUT4
         LDA  #$40
         STA  L.HB
         LDA  #$F0
         STA  L.LB
HAUT3    DEC  CURS.LIG
         JSR  AFF.
         STA  $C010
         JMP  WAIT.KBD
HAUT4    DEC  CURS.LIG
         JSR  AFF....
         STA  $C010
         JMP  WAIT.KBD
BAS      LDA  CURS.LIG
         CMP  #$1F
         BCC  BAS1
         JSR  SOUND
         STA  $C010
         JMP  WAIT.KBD
BAS1     INC  CURS.LIG
         LDA  CURS.LIG
         ASL
         ASL
         ASL
         ASL
         BCC  BAS3
         CMP  L.LB
         BNE  BAS3
         CLC
         ADC  #$10
         STA  L.LB
         BNE  BAS2
         LDA  #$41
         STA  L.HB
BAS2     JSR  AFF.
         STA  $C010
         JMP  WAIT.KBD
BAS3     JSR  AFF....
         STA  $C010
         JMP  WAIT.KBD
*******************************
*.DEPLACEMENTS.SUR.LE.DISQUE.*
*******************************
GAUCHE.  LDA  DISK.H
         BNE  GAUCHE.1
         LDA  DISK.L
         BNE  GAUCHE.1
         JSR  SOUND
         STA  $C010
         JMP  WAIT.KBD
GAUCHE.1 SEC
         LDA  DISK.L
         SBC  #$01
         STA  DISK.L
         LDA  DISK.H
         SBC  #$00
         STA  DISK.H
         JSR  AFF..
         STA  $C010
         JMP  WAIT.KBD
DROITE.  LDA  DISK.H
         BEQ  DROITE.1
         LDA  DISK.L
         CMP  #$17
         BCC  DROITE.1
         JSR  SOUND
         STA  $C010
         JMP  WAIT.KBD
DROITE.1 CLC
         LDA  DISK.L
         ADC  #$01
         STA  DISK.L
         LDA  DISK.H
         ADC  #$00
         STA  DISK.H
         JSR  AFF..
         STA  $C010
         JMP  WAIT.KBD
*****************************
*.DEPLACEMENTS.DISQUE.(*10).*
*****************************
GAUCHE.. LDA  DISK.H
         BNE  GAUCHE.2
         LDA  DISK.L
         CMP  #$0A
         BCS  GAUCHE.2
         JSR  SOUND
         STA  $C010
         JMP  WAIT.KBD
GAUCHE.2 SEC
         LDA  DISK.L
         SBC  #$0A
         STA  DISK.L
         LDA  DISK.H
         SBC  #$00
         STA  DISK.H
         JSR  AFF..
         STA  $C010
```

```
63C8- FE A5 FC C9 0F 90 33 A9
63D0- 00 85 FC A5 FD C9 1F 90
63D8- 0A A9 0F 85 FC 20 B9 69
63E0- 4C 0A 64 E6 FD A5 FD 0A
63E8- 0A 0A 0A 90 0F C5 FE D0
63F0- 0B 18 69 10 85 FE D0 04
63F8- A9 41 85 FF 20 80 66 4C
6400- 77 63 E6 FC 20 80 66 4C
6408- 77 63 20 32 67 8D 10 C0
6410- 4C 42 61 8D 10 C0 A9 06
6418- 85 07 A9 08 85 06 A9 20
6420- 85 08 20 3C 69 A2 00 AD
6428- 00 C0 10 FB 85 EF 20 B9
6430- 61 8A 48 90 0D 20 5F 67
6438- 20 B9 69 68 8D 10 C0 4C
6440- 42 61 68 AA A5 EF 29 7F
6448- 9D 10 03 C9 0D F0 20 E8
6450- 8A 48 E0 03 F0 18 A5 EF
6458- 85 08 20 3C 69 A9 20 85
6460- 08 E6 06 20 3C 69 68 AA
6468- 8D 10 C0 4C 27 64 68 A9
6470- 0D 9D 10 03 E0 00 F0 34
6478- A9 10 85 B8 A9 03 85 B9
6480- 20 67 DD 20 0C E1 A5 A0
6488- C9 01 B0 0F 85 FB A5 A1
6490- 85 FA 20 39 67 8D 10 C0
6498- 4C 42 61 C9 02 B0 0D A5
64A0- A1 C9 18 B0 07 A5 A0 85
64A8- FB 4C 8E 64 20 5F 67 20
64B0- B9 69 8D 10 C0 4C 42 61
64B8- AD 3F 6F C9 E0 F0 22 A9
64C0- E0 8D 3F 6F A9 06 85 07
64C8- A9 1B 85 06 A9 B1 85 08
64D0- 20 3C 69 E6 06 E6 06 A9
64D8- 32 85 08 20 3C 69 4C 00
64E0- 65 A9 60 8D 3F 6F A9 06
64E8- 85 07 A9 1B 85 06 A9 31
64F0- 85 08 20 3C 69 E6 06 E6
64F8- 06 A9 B2 85 08 20 3C 69
6500- 8D 10 C0 4C 42 61 20 00
6508- BF 81 3E 6F B0 06 8D 10
6510- C0 4C 42 61 20 BF 68 20
6518- 32 67 8D 10 C0 4C 42 61
6520- A5 1C F0 07 A9 00 85 1C
6528- 4C 2F 65 A9 01 85 1C 20
6530- 32 67 8D 10 C0 4C 42 61
6538- 20 B9 69 20 B9 69 A9 15
6540- 20 ED FD A2 03 BD 3A 6F
```

```
        JMP  WAIT.KBD
DROITE. LDA  DISK.H
        BEQ  DROITE.2
        LDA  DISK.L
        CMP  #$0E
        BCC  DROITE.2
        JSR  SOUND
        STA  $C010
        JMP  WAIT.KBD
DROITE.2 CLC
        LDA  DISK.L
        ADC  #$0A
        STA  DISK.L
        LDA  DISK.H
        ADC  #$00
        STA  DISK.H
        JSR  AFF..
        STA  $C010
        JMP  WAIT.KBD
*******************
*.CHANGER.DE.PAGE.*
*******************
SPACE   LDA  L.HB
        CMP  #$41
        BCS  SPACE2
        LDA  #$41
        STA  L.HB
        LDA  #$00
        STA  L.LB
        LDA  CURS.LIG
        CMP  #$10
        BCS  SPACE1
        CLC
        LDA  #$10
        ADC  CURS.LIG
        STA  CURS.LIG
SPACE1  JSR  AFF.
        STA  $C010
        JMP  WAIT.KBD
SPACE2  LDA  #$40
        STA  L.HB
        LDA  #$00
        STA  L.LB
        SEC
        LDA  CURS.LIG
        SBC  #$10
        STA  CURS.LIG
        JSR  AFF.
        STA  $C010
        JMP  WAIT.KBD
*************************
*.ENTRER.DES.CARACTERES.*
*************************
C.A.    STA  $C010
C.A.0   JSR  AFF.CUR
        LDA  CURS.COL
        LDX  #$03
C.A.1   DEX
        BEQ  C.A.2
        CLC
        ADC  CURS.COL
        JMP  C.A.1
C.A.2   CLC
        ADC  #$09
        STA  COL
        LDA  CURS.LIG
        ASL
        ASL
        ASL
        ASL
        SEC
        SBC  L.LB
        LSR
        LSR
        LSR
        LSR
        CLC
        ADC  #$08
        STA  LIG
        LDA  #$3F
        STA  CAR
        JSR  COL.LIG
        INC  COL
        JSR  COL.LIG
C.A.3   LDA  $C000
        BPL  C.A.3
        STA  $C010
        CMP  #$81
        BEQ  C.A.7
        STA  INTER
        LDA  CURS.LIG
        ASL
        ASL
        ASL
        ASL
        CLC
        ADC  CURS.COL
        SEC
        SBC  L.LB
        TAY
        LDA  INTER
        STA  (L.LB),Y
        LDA  CURS.COL
        CMP  #$0F
        BCC  C.A.6
        LDA  #$00
        STA  CURS.COL
        LDA  CURS.LIG
        CMP  #$1F
        BCC  C.A.4
        LDA  #$0F
        STA  CURS.COL
        JSR  SOUND
        JMP  C.A.7
C.A.4   INC  CURS.LIG
        LDA  CURS.LIG
        ASL
        ASL
        ASL
        ASL
        ASL
        BCC  C.A.5
        CMP  L.LB
        BNE  C.A.5
        CLC
        ADC  #$10
        STA  L.LB
        BNE  C.A.5
        LDA  #$41
        STA  L.HB
C.A.5   JSR  AFF
        JMP  C.A.0
C.A.6   INC  CURS.COL
        JSR  AFF
        JMP  C.A.0
C.A.7   JSR  AFF.
        STA  $C010
        JMP  WAIT.KBD
*******************
*.ENTRER.UN.BLOCK.*
*******************
C.B.    STA  $C010
        LDA  #$06
        STA  LIG
        LDA  #$08
        STA  COL
        LDA  #$20
        STA  CAR
```

```
6548- 95 36 CA 10 F8 20 58 FC
6550- 20 B9 69 20 B9 69 8D 10
6558- C0 60 8D 10 C0 85 08 A5
6560- FC A2 03 CA F0 06 18 65
6568- FC 4C 63 65 18 69 09 85
6570- 06 A5 FD 0A 0A 0A 0A 38
6578- E5 FE 4A 4A 4A 4A 18 69
6580- 08 85 07 20 3C 69 A5 08
6588- C9 C1 B0 06 38 E9 B0 4C
6590- 95 65 38 E9 B7 0A 0A 0A
6598- 0A 85 EF AD 00 C0 10 FB
65A0- 20 A5 61 B0 0C 8D 10 C0
65A8- 20 32 67 20 B9 69 4C 42
65B0- 61 C9 C1 B0 06 38 E9 B0
65B8- 4C BE 65 38 E9 B7 05 EF
65C0- 85 EF A5 FD 0A 0A 0A 0A
65C8- 18 65 FC 38 E5 FE A8 A5
65D0- EF 91 FE 8D 10 C0 20 32
65D8- 67 4C 42 61 A9 5A 85 08
65E0- 20 3C 69 A9 4C 85 08 E6
65E8- 06 20 3C 69 E6 06 A9 4D
65F0- C5 06 D0 F5 A9 5F 85 08
65F8- 20 3C 69 E6 06 A9 A0 85
6600- 08 20 3C 69 E6 06 A9 5A
6608- 85 08 20 3C 69 E6 07 A9
6610- 00 85 06 A9 5A 85 08 20
6618- 3C 69 E6 06 A9 A0 85 08
6620- 20 3C 69 E6 06 A9 A4 85
6628- 08 20 3C 69 A9 5F 85 08
6630- A9 4D 85 06 20 3C 69 E6
6638- 06 E6 06 A9 5A 85 08 20
6640- 3C 69 A9 17 C5 07 D0 C5
6648- E6 07 A9 00 85 06 A9 5A
6650- 85 08 20 3C 69 A9 DF 85
6658- 08 E6 06 20 3C 69 E6 06
6660- A9 4D C5 06 D0 F5 A9 5F
6668- 85 08 20 3C 69 E6 06 A9
6670- A0 85 08 20 3C 69 E6 06
6678- A9 52 85 08 20 3C 69 60
6680- A5 FE 85 1A A5 FF 85 1B
6688- A9 07 48 20 C1 FB A9 03
6690- 85 24 A5 1B 29 01 F0 05
6698- A9 B1 4C 9F 66 A9 B0 20
66A0- ED FD A5 1A 20 DA FD A9
66A8- A0 20 ED FD A9 A0 20 ED
66B0- FD A9 00 48 A9 A0 20 ED
66B8- FD 68 48 A8 B1 1A 20 DA
66C0- FD 68 A8 C8 C0 10 98 90
```

```
        JSR  COL.LIG
        LDX  #$00
C.B.1   LDA  $C000
        BPL  C.B.1
        STA  INTER
        JSR  TEST..
        TXA
        PHA
        BCC  C.B.2
        JSR  AFF...
        JSR  SOUND
        PLA
        STA  $C010
        JMP  WAIT.KBD.
C.B.2   PLA
        TAX
        LDA  INTER
        AND  #$7F
        STA  $310,X
        CMP  #$0D
        BEQ  C.B.4
        INX
        TXA
        PHA
        CPX  #$03
        BEQ  C.B.3
        LDA  INTER
        STA  CAR
        JSR  COL.LIG
        LDA  #$20
        STA  CAR
        INC  COL
        JSR  COL.LIG
        PLA
        TAX
        STA  $C010
        JMP  C.B.1
C.B.3   PLA
C.B.4   LDA  #$0D
        STA  $310,X
        CPX  #$00
        BEQ  C.B.7
        LDA  #$10
        STA  $B8
        LDA  #$03
        STA  $B9
        JSR  FRMNUM
        JSR  AYINT
        LDA  $A0
        CMP  #$01
        BCS  C.B.6
        STA  DISK.H
C.B.5   LDA  $A1
        STA  DISK.L
        JSR  AFF..
        STA  $C010
        JMP  WAIT.KBD
C.B.6   CMP  #$02
        BCS  C.B.7
        LDA  $A1
        CMP  #$18
        BCS  C.B.7
        LDA  $A0
        STA  DISK.H
        JMP  C.B.5
C.B.7   JSR  AFF...
        JSR  SOUND
        STA  $C010
        JMP  WAIT.KBD
********************
*.CHANGER.DE.DRIVE.*
********************
C.C.    LDA  TABLE+1
        CMP  #$E0
        BEQ  C.C.1
        LDA  #$E0
        STA  TABLE+1
        LDA  #$06
        STA  LIG
        LDA  #$1B
        STA  COL
        LDA  #$B1
        STA  CAR
        JSR  COL.LIG
        INC  COL
        INC  COL
        LDA  #$32
        STA  CAR
        JSR  COL.LIG
        JMP  C.C.2
C.C.1   LDA  #$60
        STA  TABLE+1
        LDA  #$06
        STA  LIG
        LDA  #$1B
        STA  COL
        LDA  #$31
        STA  CAR
        JSR  COL.LIG
        INC  COL
        INC  COL
        LDA  #$B2
        STA  CAR
        JSR  COL.LIG
C.C.2   STA  $C010
        JMP  WAIT.KBD
************************
*.SAUVER.SUR.DISQUETTE.*
************************
C.S.    JSR  $BF00
        DFB  $81
        DA   TABLE
        BCS  ERREUR.S
        STA  $C010
        JMP  WAIT.KBD
ERREUR.S JSR ERROR
        JSR  AFF.
        STA  $C010
        JMP  WAIT.KBD
***************************
*."SWITCHER".CARACTERES.*
***************************
C.X.    LDA  SWITCH
        BEQ  C.X.1
        LDA  #$00
        STA  SWITCH
        JMP  C.X.2
C.X.1   LDA  #$01
        STA  SWITCH
C.X.2   JSR  AFF.
        STA  $C010
        JMP  WAIT.KBD
************************
*.SORTIR.DU.PROGRAMME.*
************************
ESC     JSR  SOUND
        JSR  SOUND
        LDA  #21
        JSR  COUT
        LDX  #3
RESETIO LDA  SAVE,X
        STA  $36,X
        DEX
        BPL  RESETIO
        JSR  HOME
        JSR  SOUND
        JSR  SOUND
```

```
66C8- EA A9 A0 20 ED FD A9 A0
66D0- 20 ED FD A9 BA 20 ED FD
66D8- A9 A0 20 ED FD A9 3C 85
66E0- 06 68 48 AA E8 8A 85 07
66E8- A9 00 48 A8 B1 1A 85 EF
66F0- A5 1C F0 0F A5 EF F0 07
66F8- 09 80 85 EF 4C 03 67 A9
6700- AE 85 EF A5 EF 85 08 38
6708- 66 09 20 3C 69 26 09 E6
6710- 06 68 A8 C8 98 C0 10 90
6718- D1 68 A8 C8 C0 17 B0 11
6720- 18 A5 1A 69 10 85 1A A9
6728- 00 65 1B 85 1B 98 4C 8A
6730- 66 60 20 80 66 20 28 68
6738- 60 A5 FA 8D 42 6F A5 FB
6740- 8D 43 6F 20 00 BF 80 3E
6748- 6F B0 07 20 5F 67 20 32
6750- 67 60 20 A0 68 20 BF 68
6758- 20 5F 67 20 32 67 60 A5
6760- 1E 85 07 A9 4F 85 06 A9
6768- 5A 85 08 20 3C 69 A5 FB
6770- A4 FA 20 F2 E2 20 63 EB
6778- A9 00 A0 0E 20 F2 E2 20
6780- 69 EA 20 63 EB A9 00 A0
6788- 04 20 F2 E2 20 C1 E7 20
6790- 0C E1 A5 A1 85 1E 85 07
6798- A9 20 85 08 20 3C 69 A9
67A0- 05 20 C1 FB A9 08 85 24
67A8- A6 FA A5 FB 20 24 ED A5
67B0- FB D0 14 A5 FA C9 64 B0
67B8- 0E C9 0A B0 05 A9 A0 20
67C0- ED FD A9 A0 20 ED FD A9
67C8- 0E 85 24 A5 FB F0 05 A9
67D0- B1 4C D6 67 A9 B0 20 ED
67D8- FD A5 FA 20 DA FD 60 A9
67E0- 45 6F A2 03 CA F0 07 18
67E8- 6D 45 6F 4C E4 67 18 69
67F0- 08 85 24 AD 46 6F 0A 0A
67F8- 0A 0A 38 E5 FE 4A 4A 4A
6800- 4A 18 69 07 20 C1 FB A9
6808- A0 20 ED FD AD 46 6F 0A
6810- 0A 0A 0A 18 6D 45 6F 38
6818- E5 FE A8 B1 FE 20 DA FD
6820- A9 A0 20 ED FD 4C 28 68
6828- A5 FC 8D 45 6F A5 FD 8D
6830- 46 6F A9 28 85 24 A9 05
6838- 20 C1 FB A5 FD 0A 0A 0A
6840- 0A 85 EF 90 05 A9 B1 4C
```

```
        STA  $C010
        RTS
*************************
*.ENTRER.UNE.VALEUR.HEXA.*
*************************
ENT.CAR STA  $C010
        STA  CAR
        LDA  CURS.COL
        LDX  #$03
ENT.CAR1 DEX
        BEQ  ENT.CAR2
        CLC
        ADC  CURS.COL
        JMP  ENT.CAR1
ENT.CAR2 CLC
        ADC  #$09
        STA  COL
        LDA  CURS.LIG
        ASL
        ASL
        ASL
        ASL
        SEC
        SBC  L.LB
        LSR
        LSR
        LSR
        LSR
        CLC
        ADC  #$08
        STA  LIG
        JSR  COL.LIG
        LDA  CAR
        CMP  #$C1
        BCS  ENT.CAR3
        SEC
        SBC  #$B0
        JMP  ENT.CAR4
ENT.CAR3 SEC
        SBC  #$B7
ENT.CAR4 ASL
        ASL
        ASL
        ASL
        STA  INTER
ENT.CAR5 LDA  $C000
        BPL  ENT.CAR5
        JSR  TEST
        BCS  ENT.CAR6
        STA  $C010
        JSR  AFF.
        JSR  SOUND
        JMP  WAIT.KBD
ENT.CAR6 CMP  #$C1
        BCS  ENT.CAR7
        SEC
        SBC  #$B0
        JMP  ENT.CAR8
ENT.CAR7 SEC
        SBC  #$B7
ENT.CAR8 ORA  INTER
        STA  INTER
        LDA  CURS.LIG
        ASL
        ASL
        ASL
        ASL
        CLC
        ADC  CURS.COL
        SEC
        SBC  L.LB
        TAY
        LDA  INTER
        STA  (L.LB),Y
        STA  $C010
        JSR  AFF.
        JMP  WAIT.KBD
****************************
*.TRACE.DU.CADRE.ENTOURANT.*
*.LE.CONTENU.DU.BUFFER.....*
****************************
CADRE   LDA  #$5A       ;"Z"
        STA  CAR
        JSR  COL.LIG
        LDA  #$4C       ;"L"
        STA  CAR
        INC  COL
CADRE.1 JSR  COL.LIG
        INC  COL
        LDA  #77
        CMP  COL
        BNE  CADRE.1
        LDA  #$5F       ;"_"
        STA  CAR
        JSR  COL.LIG
        INC  COL
        LDA  #$A0       ;" "
        STA  CAR
        JSR  COL.LIG
        INC  COL
        LDA  #$5A       ;"Z"
        STA  CAR
        JSR  COL.LIG
CADRE.2 INC  LIG
        LDA  #$00
        STA  COL
        LDA  #$5A       ;"Z"
        STA  CAR
        JSR  COL.LIG
        INC  COL
        LDA  #$A0       ;" "
        STA  CAR
        JSR  COL.LIG
        INC  COL
        LDA  #$A4       ;"$"
        STA  CAR
        JSR  COL.LIG
        LDA  #$5F       ;"_"
        STA  CAR
        LDA  #77
        STA  COL
        JSR  COL.LIG
        INC  COL
        INC  COL
        LDA  #$5A       ;"Z"
        STA  CAR
        JSR  COL.LIG
        LDA  #23
        CMP  LIG
        BNE  CADRE.2
        INC  LIG
        LDA  #$00
        STA  COL
        LDA  #$5A       ;"Z"
        STA  CAR
        JSR  COL.LIG
        LDA  #$DF       ;"_"
        STA  CAR
        INC  COL
CADRE.3 JSR  COL.LIG
        INC  COL
        LDA  #77
        CMP  COL
        BNE  CADRE.3
        LDA  #$5F       ;"_"
        STA  CAR
        JSR  COL.LIG
```

```
6848- 4C 68 A9 B0 20 ED FD A5
6850- FC 05 EF 20 DA FD A5 FC
6858- A2 03 CA F0 06 18 65 FC
6860- 4C 5A 68 18 69 08 85 24
6868- A5 FD 0A 0A 0A 0A 38 E5
6870- FE 4A 4A 4A 4A 18 69 07
6878- 20 C1 FB A9 BE 20 ED FD
6880- A9 3F 85 32 A5 FD 0A 0A
6888- 0A 0A 18 65 FC 38 E5 FE
6890- A8 B1 FE 20 DA FD A9 FF
6898- 85 32 A9 BC 20 ED FD 60
68A0- A9 40 85 EC A9 00 85 EB
68A8- A0 00 A9 00 91 EB C8 D0
68B0- F9 A9 41 85 EC A0 00 A9
68B8- 00 91 EB C8 D0 F9 60 A9
68C0- 08 85 07 20 37 69 A9 83
68C8- A0 6E 20 00 6A 20 37 69
68D0- A9 9F A0 6E 20 00 6A 20
68D8- 37 69 A9 BB A0 6E 20 00
68E0- 6A 20 37 69 A9 9F A0 6E
68E8- 20 00 6A 20 37 69 A9 D7
68F0- A0 6E 20 00 6A A9 EA A0
68F8- 6E 24 09 30 04 A9 F4 A0
6900- 6E 20 00 6A 20 37 69 A9
6908- 9F A0 6E 20 00 6A 20 37
6910- 69 A9 FE A0 6E 20 00 6A
6918- 20 37 69 20 B9 69 20 B9
6920- 69 8D 10 C0 AD 00 C0 10
6928- FB C9 8D F0 09 8D 10 C0
6930- 20 B9 69 4C 24 69 60 A9
6938- 1B 85 06 60 A5 07 20 80
6940- 69 A5 06 4A B0 10 8D 01
6948- C0 8D 55 C0 20 56 69 8D
6950- 54 C0 8D 00 C0 60 A8 A5
6958- 08 24 09 30 20 C9 5A F0
6960- 04 C9 5F D0 02 A9 A1 C9
6968- 4C F0 04 C9 DF D0 02 A9
6970- AD C9 51 D0 02 A9 F6 C9
6978- 52 D0 02 A9 DE 91 EB 60
6980- C9 09 B0 07 A0 00 84 EB
6988- 4C A0 69 C9 11 B0 0A A0
6990- 28 84 EB 38 E9 08 4C A0
6998- 69 A0 50 84 EB 38 E9 10
69A0- A0 04 84 EC AA CA F0 10
69A8- 18 A9 80 65 EB 85 EB A9
69B0- 00 65 EC 85 EC CA D0 F0
69B8- 60 A2 FF A9 58 A0 1B 88
69C0- D0 FD 2C 30 C0 A8 88 D0
```

```
        INC  COL
        LDA  #$A0        ;" "
        STA  CAR
        JSR  COL.LIG
        INC  COL
        LDA  #$52        ;"R"
        STA  CAR
        JSR  COL.LIG
        RTS
*****************************
*.AFFICHER.CONTENU.DU.BUFFER.*
*****************************
AFF     LDA  L.LB
        STA  AFFI.L
        LDA  L.HB
        STA  AFFI.H
        LDA  #$07        ;LIG8
AFF0    PHA
        JSR  BASCALC
        LDA  #$03        ;COL3
        STA  $24         ;CH
        LDA  AFFI.H
        AND  #$01
        BEQ  AFF1
        LDA  #$B1        ;"1"
        JMP  AFF2
AFF1    LDA  #$B0        ;"0"
AFF2    JSR  COUT
        LDA  AFFI.L
        JSR  PRBYTE
        LDA  #$A0
        JSR  COUT
        LDA  #$A0
        JSR  COUT
        LDA  #$00
AFF3    PHA
        LDA  #$A0
        JSR  COUT
        PLA
        PHA
        TAY
        LDA  (AFFI.L),Y
        JSR  PRBYTE
        PLA
        TAY
        INY
        CPY  #16
        TYA
        BCC  AFF3
        LDA  #$A0
        JSR  COUT
        LDA  #$A0
        JSR  COUT
        LDA  #$BA        ;":"
        JSR  COUT
        LDA  #$A0
        JSR  COUT
        LDA  #60
        STA  COL
        PLA
        PHA
        TAX
        INX
        TXA
        STA  LIG
        LDA  #$00
AFF4    PHA
        TAY
        LDA  (AFFI.L),Y
        STA  INTER
        LDA  SWITCH
        BEQ  AFF6
        LDA  INTER
        BEQ  AFF5
        ORA  #$80
        STA  INTER
        JMP  AFF6
AFF5    LDA  #"."
        STA  INTER
AFF6    LDA  INTER
        STA  CAR
        SEC              ;++
        ROR  ICONE       ;++
        JSR  COL.LIG
        ROL  ICONE       ;++
        INC  COL
        PLA
        TAY
        INY
        TYA
        CPY  #16
        BCC  AFF4
        PLA
        TAY
        INY
        CPY  #23
        BCS  AFF7
        CLC
        LDA  AFFI.L
        ADC  #16
        STA  AFFI.L
        LDA  #$00
        ADC  AFFI.H
        STA  AFFI.H
        TYA
        JMP  AFF0
AFF7    RTS
**************************
*.AFFICHER.BUFFER+CURSEUR.*
**************************
AFF.    JSR  AFF
        JSR  AFF.CUR
        RTS
*******************************
*.CHARGER.ET.AFFICHER.UN.BLOCK.*
*******************************
AFF..   LDA  DISK.L
        STA  TABLE+4
        LDA  DISK.H
        STA  TABLE+5
        JSR  $BF00
        DFB  $80
        DA   TABLE
        BCS  ERREUR
        JSR  AFF...
        JSR  AFF.
        RTS
ERREUR  JSR  EFFACAGE
        JSR  ERROR
        JSR  AFF...
        JSR  AFF.
        RTS
****************************
*.AFFICHER.NUMERO.DU.BLOCK.*
****************************
AFF...  LDA  CURS
        STA  LIG
        LDA  #79
        STA  COL
        LDA  #$5A        ;"Z"
        STA  CAR
        JSR  COL.LIG
        LDA  DISK.H
        LDY  DISK.L
        JSR  GIVAYF
        JSR  MOVAF
        LDA  #$00
```

```
69C8- FD E9 01 F0 EE 2C 30 C0
69D0- CA D0 EA 60 85 ED 84 EE
69D8- A0 00 98 48 B1 ED D0 02
69E0- 68 60 20 EB 69 68 A8 C8
69E8- 4C DA 69 C9 8D F0 0A 85
69F0- 08 20 3C 69 E6 06 E6 06
69F8- 60 E6 07 A9 01 85 06 60
6A00- 85 ED 84 EE A0 00 98 48
6A08- B1 ED D0 02 68 60 20 17
6A10- 6A 68 A8 C8 4C 06 6A C9
6A18- 8D F0 08 85 08 20 3C 69
6A20- E6 06 60 E6 07 A9 00 85
6A28- 06 60 8D 8D 8D 8D A0 A0
6A30- A0 A0 A0 A0 A0 A0 DF DF
6A38- DF DF DF DF DF DF DF DF
6A40- DF DF DF DF DF DF DF DF
6A48- DF DF DF DF DF 8D A0 A0
6A50- A0 A0 A0 A0 A0 5A A0 A0
6A58- A0 A0 A0 A0 A0 A0 A0 A0
6A60- A0 A0 A0 A0 A0 A0 A0 A0
6A68- A0 A0 A0 A0 A0 5F 8D A0
6A70- A0 A0 A0 A0 A0 A0 5A A0
6A78- A0 A0 A0 A0 A0 D2 AE D3
6A80- AE D0 AE C4 AE BA A0 A0
6A88- A0 A0 A0 A0 A0 A0 5F 8D
6A90- 00 A0 A0 A0 A0 A0 A0 A0
6A98- 5A A0 A0 A0 A0 A0 A0 4C
6AA0- 4C 4C 4C 4C 4C 4C 4C 4C
6AA8- A0 A0 A0 A0 A0 A0 A0 A0
6AB0- 5F 8D A0 A0 A0 A0 A0 A0
6AB8- A0 5A A0 A8 D2 E5 E1 E4
6AC0- A0 A6 A0 D3 E1 F6 E5 A0
6AC8- D0 F2 EF C4 EF F3 A9 A0
6AD0- A0 5F 8D A0 A0 A0 A0 A0
6AD8- A0 A0 5A A0 A0 A0 A0 A0
6AE0- A0 A0 A0 A0 A0 A0 A0 A0
6AE8- A0 A0 A0 A0 A0 A0 A0 A0
6AF0- A0 A0 5F 8D 00 A0 A0 A0
6AF8- A0 A0 A0 A0 5A A0 D5 F4
6B00- E9 EC E9 F4 E1 E9 F2 E5
6B08- A0 E4 E5 A0 CC E5 E3 F4
6B10- F5 F2 E5 A0 5F 8D A0 A0
6B18- A0 A0 A0 A0 A0 5A A0 E5
6B20- F4 A0 E4 A7 E5 E3 F2 E9
6B28- F4 F5 F2 E5 A0 E4 E9 F2
6B30- E5 E3 F4 E5 A0 5F 8D A0
6B38- A0 A0 A0 A0 A0 A0 5A A0
6B40- F3 F5 F2 A0 C4 E9 F3 F1
```

```
        LDY  #14
        JSR  GIVAYF
        JSR  FDIVT
        JSR  MOVAF
        LDA  #$00
        LDY  #4
        JSR  GIVAYF
        JSR  FADDT
        JSR  AYINT
        LDA  $A1
        STA  CURS
        STA  LIG
        LDA  #$20
        STA  CAR
        JSR  COL.LIG
        LDA  #$05        ;LIG6
        JSR  BASCALC
        LDA  #$08        ;COL8
        STA  $24         ;CH
        LDX  DISK.L
        LDA  DISK.H
        JSR  LINPTR
        LDA  DISK.H
        BNE  AFF...2
        LDA  DISK.L
        CMP  #$64
        BCS  AFF...2
        CMP  #$0A
        BCS  AFF...1
        LDA  #$A0
        JSR  COUT
AFF...1 LDA  #$A0
        JSR  COUT
AFF...2 LDA  #$0E
        STA  $24         ;CH
        LDA  DISK.H
        BEQ  AFF...3
        LDA  #"1"
        JMP  AFF...4
AFF...3 LDA  #"0"
AFF...4 JSR  COUT
        LDA  DISK.L
        JSR  PRBYTE
        RTS
*******************************
*.EFFACER.ET.AFFICHER.CURSEUR.*
*******************************
AFF.... LDA  TABLE1      ;COL
        LDX  #$03
AFF....1 DEX
         BEQ  AFF....2
         CLC
         ADC  TABLEI
         JMP  AFF....1
AFF....2 CLC
         ADC  #$08
         STA  $24        ;CH
         LDA  TABLE1+I   ;LIG
         ASL
         ASL
         ASL
         ASL
         SEC
         SBC  L.LB
         LSR
         LSR
         LSR
         LSR
         CLC
         ADC  #$07
         JSR  BASCALC
         LDA  #$A0
         JSR  COUT
         LDA  TABLE1+1
         ASL
         ASL
         ASL
         ASL
         CLC
         ADC  TABLE1
         SEC
         SBC  L.LB
         TAY
         LDA  (L.LB),Y
         JSR  PRBYTE
         LDA  #$A0
         JSR  COUT
         JMP  AFF.CUR
************************
*.AFFICHER.LE.CURSEUR.*
************************
AFF.CUR  LDA  CURS.COL
         STA  TABLE1
         LDA  CURS.LIG
         STA  TABLE1+1
         LDA  #40
         STA  $24        ;CH
         LDA  #$05
         JSR  BASCALC
         LDA  CURS.LIG
         ASL
         ASL
         ASL
         ASL
         ASL
         STA  INTER
         BCC  AFF.X1
         LDA  #"1"
         JMP  AFF.X2
AFF.X1   LDA  #"0"
AFF.X2   JSR  COUT
         LDA  CURS.COL
         ORA  INTER
         JSR  PRBYTE
         LDA  CURS.COL
         LDX  #$03
AFF.CUR1 DEX
         BEQ  AFF.CUR2
         CLC
         ADC  CURS.COL
         JMP  AFF.CUR1
AFF.CUR2 CLC
         ADC  #$08
         STA  $24        ;CH
         LDA  CURS.LIG
         ASL
         ASL
         ASL
         ASL
         ASL
         SEC
         SBC  L.LB
         LSR
         LSR
         LSR
         LSR
         CLC
         ADC  #$07
         JSR  BASCALC
         LDA  #">"
         JSR  COUT
         LDA  #$3F
         STA  $32
         LDA  CURS.LIG
         ASL
         ASL
         ASL
         ASL
         CLC
```

```
6B48- F5 E5 F4 F4 E5 F3 A0 D0
6B50- F2 EF C4 CF D3 A0 5F 8D
6B58- A0 A0 A0 A0 A0 A0 A0 5A
6B60- A0 A0 A0 A0 A0 A0 A0 A0
6B68- AD A0 A0 A0 A0 A0 A0 A0
6B70- A0 A0 A0 A0 A0 A0 A0 5F
6B78- 8D 00 A0 AD A0 A0 A0 A0
6B80- A0 5A A0 A0 A0 F0 E1 F2
6B88- A0 CD E1 F2 E3 A0 D2 CF
6B90- C7 CC C9 C1 CE CF AD A0
6B98- A0 5F 8D A0 A0 A0 A0 A0
6BA0- A0 A0 5A A0 A0 A0 A0 A0
6BA8- A0 A0 A0 A0 A0 A0 A0 A0
6BB0- A0 A0 AD AD A0 A0 A0 AD
6BB8- A0 A0 5F 8D AD A0 A0 A0
6BC0- A0 AD A0 A0 4C 4C 4C 4C
6BC8- 4C 4C 4C 4C 4C 4C 4C 4C
6BD0- 4C 4C 4C 4C 4C 4C 4C 4C
6BD8- 4C 4C 4C 8D 8D 8D 8D A0
6BE0- A0 A0 A0 A0 A0 A0 A0 A0
6BE8- AD A0 A0 A0 A0 A0 C1 F0
6BF0- F0 F5 F9 E5 FA A0 F3 F5
6BF8- F2 AD 00 4D AD 4E 8D A0
6C00- A0 AD A0 A0 A0 AD A0 A0
6C08- A0 A0 A0 A0 A0 A0 4C 4C
6C10- 4C 4C 4C 4C 4C 4C 4C 4C
6C18- 4C 4C 4C 8D 00 D2 C5 D4
6C20- D5 D2 CE A0 20 8D A0 A0
6C28- A0 A0 A0 A0 A0 A0 A0 A0
6C30- A0 A0 A0 A0 A0 4C 4C 4C
6C38- 4C 4C 4C 4C 4C 4C 4C 4C
6C40- 4C 4C 4C 4C 4C 4C 4C 8D
6C48- 00 D2 AE D3 AE D0 AE C4
6C50- AE BA A0 D5 F4 E9 EC E9
6C58- F4 E1 E9 F2 E5 A0 E4 E5
6C60- A0 CC E5 E3 F4 F5 F2 E5
6C68- A0 E5 F4 A0 E4 A7 C5 E3
6C70- F2 E9 F4 F5 F2 E5 A0 F3
6C78- F5 F2 A0 E4 E9 F3 EB F3
6C80- AD D0 F2 EF C4 CF D3 A0
6C88- F0 E1 F2 A0 CD E1 F2 E3
6C90- A0 D2 CF C7 CC C9 C1 CE
6C98- CF 8D A0 A0 C3 EF ED ED
6CA0- E1 EE E4 E5 F3 BA 5A A0
6CA8- 00 48 A0 55 A0 4B A0 4A
6CB0- 00 E6 EC E5 E3 E8 E5 F3
6CB8- 00 A0 BA AD C4 E5 F0 EC
6CC0- E1 E3 E5 ED E5 EE F4 F3
```

```
        ADC  CURS.COL
        SEC
        SBC  L.LB
        TAY
        LDA  (L.LB),Y
        JSR  PRBYTE
        LDA  #$FF        ;NORMAL
        STA  $32
        LDA  #"<"
        JSR  COUT
        RTS
*********************
*.EFFACER.LE.BUFFER.*
*********************
EFFACAGE LDA  #$40
        STA  HB
        LDA  #$00
        STA  LB
        LDY  #$00
EFF.1   LDA  #$00
        STA  (LB),Y
        INY
        BNE  EFF.1
        LDA  #$41
        STA  HB
        LDY  #$00
EFF.2   LDA  #$00
        STA  (LB),Y
        INY
        BNE  EFF.2
        RTS
*****************************
*.ROUTINE.DE.TRAITEMENT.DES.*
*.ERREURS.DU.DISQUE........*
*****************************
ERROR   LDA  #$08
        STA  LIG
        JSR  ERROR.
        LDA  #<ERROR.0
        LDY  #>ERROR.0
        JSR  STROUT80
        JSR  ERROR.
        LDA  #<ERROR.1
        LDY  #>ERROR.1
        JSR  STROUT80
        JSR  ERROR.
        LDA  #<ERROR.2
        LDY  #>ERROR.2
        JSR  STROUT80
        JSR  ERROR.
        LDA  #<ERROR.3
        LDY  #>ERROR.3
        JSR  STROUT80
        LDA  #<ERROR4.1
        LDY  #>ERROR4.1
        BIT  ICONE
        BMI  ERICON
        LDA  #<ERROR4.2
        LDY  #>ERROR4.2
ERICON  JSR  STROUT80
        JSR  ERROR.
        LDA  #<ERROR.1
        LDY  #>ERROR.1
        JSR  STROUT80
        JSR  ERROR.
        LDA  #<ERROR.5
        LDY  #>ERROR.5
        JSR  STROUT80
        JSR  ERROR.
        JSR  SOUND
        JSR  SOUND
        STA  $C010
ERROR0  LDA  $C000
        BPL  ERROR0
        CMP  #$8D
        BEQ  ERROR1
        STA  $C010
        JSR  SOUND
        JMP  ERROR0
ERROR1  RTS
ERROR.  LDA  #27
        STA  COL
        RTS
*****************************
**........................**
**.UTILITAIRE.LIGNE.COLONNE.**
**........................**
*****************************
COL.LIG LDA  LIG
        JSR  CALCUL
        LDA  COL
        LSR
        BCS  COLONNE
        STA  STO80.1
        STA  PAGE2
        JSR  COLONNE
        STA  PAGE1
        STA  STO80.0
        RTS
COLONNE TAY
        LDA  CAR
        BIT  ICONE
        BMI  X6
        CMP  #$5A
        BEQ  X1
        CMP  #$5F
        BNE  X2
X1      LDA  #"!"
X2      CMP  #$4C
        BEQ  X3
        CMP  #$DF
        BNE  X4
X3      LDA  #"-"
X4      CMP  #$51
        BNE  X5
        LDA  #"v"
X5      CMP  #$52
        BNE  X6
        LDA  #$DE
X6      EQU  *
        STA  (LB),Y
        RTS
CALCUL  CMP  #$09
        BCS  CAL1
        LDY  #$00
        STY  LB
        JMP  SUITE
CAL1    CMP  #$11
        BCS  CAL2
        LDY  #$28
        STY  LB
        SEC
        SBC  #$08
        JMP  SUITE
CAL2    LDY  #$50
        STY  LB
        SEC
        SBC  #$10
SUITE   LDY  #$04
        STY  HB
        TAX
        DEX
        BEQ  END
```

```
6CC8- AD F3 F5 F2 A0 EC A7 E5
6CD0- E3 F2 E1 EE A0 5F DE C1
6CD8- A0 BA A0 C5 EE F4 F2 E5
6CE0- F2 A0 E3 E1 F2 E1 E3 F4
6CE8- E5 F2 E5 F3 A0 A0 A0 A0
6CF0- 5F 8D A0 A0 4C 4C 4C 4C
6CF8- 4C 4C 4C 4C 4C A0 5A A0
6D00- 00 40 A0 48 A0 55 A0 BA
6D08- A0 C4 E5 F0 EC E1 E3 E5
6D10- ED E5 EE F4 F3 A0 F3 F5
6D18- F2 A0 C4 E9 F3 F1 F5 E5
6D20- F4 F4 E5 A0 00 D0 EF ED
6D28- ED E5 A0 E6 E5 F2 ED E5
6D30- E5 A0 BC AD A0 AD BE A0
6D38- BA A0 E0 E0 A0 F3 F5 F2
6D40- A0 C4 E9 F3 F1 F5 E5 A0
6D48- 00 5F DE C2 A0 BA A0 C5
6D50- EE F4 F2 E5 F2 A0 F5 EE
6D58- A0 C2 EC EF E3 EB A0 A0
6D60- A0 A0 A0 A0 5F A0 A0 51
6D68- 8D A0 A0 A0 A8 C5 D3 C3
6D70- BA A0 A0 A0 A0 5A A0 00
6D78- 41 A0 48 A0 55 A0 BA A0
6D80- C4 E5 F0 EC E1 E3 E5 ED
6D88- E5 EE F4 F3 A0 A8 AA B1
6D90- B0 A9 A0 A0 A0 A0 A0 A0
6D98- A0 A0 A0 00 D0 EF ED ED
6DA0- E5 A0 EF F5 F6 E5 F2 F4
6DA8- E5 A0 BC AD A0 AD BE A0
6DB0- BA A0 C4 E5 F0 EC AE A0
6DB8- F8 A0 B1 B0 A0 A0 A0 00
6DC0- 5F DE C3 A0 BA A0 C3 E8
6DC8- E1 EE E7 E5 F2 A0 E4 E5
6DD0- A0 CC E5 E3 F4 E5 F5 F2
6DD8- A0 A0 A0 5F A0 A0 5A 8D
6DE0- A0 A0 A0 D1 F5 E9 F4 F4
6DE8- E5 F2 A9 A0 5A A0 A0 C5
6DF0- D3 D0 C1 C3 C5 A0 BA A0
6DF8- C3 E8 E1 EE E7 E5 F2 A0
6E00- E4 E5 A0 D0 E1 E7 E5 A0
6E08- A0 A0 A0 A0 A0 A0 A0 A0
6E10- A0 5F DE D3 A0 BA A0 D3
6E18- E1 F5 F6 E5 F2 A0 F3 F5
6E20- F2 A0 C4 E9 F3 F1 F5 E5
6E28- F4 F4 E5 A0 5F A0 A0 5A
6E30- 8D A0 C2 CC CF C3 CB BA
6E38- A0 A0 A0 A0 A0 A8 A4 A0
6E40- A0 A0 A9 A0 A0 C4 D2 C9
```

```
R.S.P.D.: Utilitaire de Lecture et d'Ecriture sur disks ProDOS par Marc ROGLIANO
  Commandes: | ← → ↑ ↓ : Deplacements sur l'ecran | ^A : Entrer caracteres
  ---------  | ⌘ ← → : Deplacements sur Disquette | ^B : Entrer un Block
   (ESC:     | ⇧ ← → : Deplacements (*10)         | ^C : Changer de Lecteur
   Quitter)  |  ESPACE : Changer de Page          | ^S : Sauver sur Disquette
 BLOCK: 2   ($002)  DRIVE: 1/2  OCTET: $113       | ^X : Switcher Caracteres

 $020   00 00 C3 27 00 05 00 06 00 18 01 26 50 52 4F 44  : ..C'ME.F.XA&PROO
 $030   4F 53 00 00 00 00 00 00 00 00 00 FF 08 00 1E 00  : OS.........>H.^.
 $040   00 3A 00 00 00 00 00 00 00 21 00 20 00 00 00 00  : .:.......!. ....
 $050   02 00 2C 42 41 53 49 43 2E 53 59 53 54 45 40 00  : B.,BASIC.SYSTEM.
 $060   00 00 FF 26 00 15 00 00 28 00 00 00 00 00 00 00  : ..>&.U..(.......
 $070   21 00 20 00 00 00 00 02 00 2E 52 2E 53 2E 50 2E  : !. ....B..R.S.P.
 $080   44 2E 53 4F 55 52 43 45 00 04 3B 00 29 00 00 50  : D.SOURCE.0;.).P
 $090   00 00 00 00 00 00 00 21 00 00 00 00 00 00 02 00  : .......!......B.
 $0A0   20 52 2E 53 2E 50 2E 44 2E 4F 42 4A 45 54 00 00  : -R.S.P.D.OBJET..
 $0B0   06 64 00 09 00 45 0F 00 94 AC 00 00 00 00 21 00  : Fd.I.E0.T,....!.
 $0C0   60 00 00 00 00 02 00 17 53 54 41 52 54 55 50 00  : `....B.WSTARTUP.
 $0D0   00 00 00 00 00 00 00 FC 6C 00 01 00 10 00 00 00  : .......ùl.A.]...
 $0E0   00 00 00 00 00 E3 01 08 00 00 00 00 02 00 00 53  : .....cAH....B..S
 $0F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 6E  : ..............Dn
 $100   00 29 00 00 50 00 00 00 00 00 00 00 E3 00 00 00  : .)..P.......c...
 $110   00 00 00>[illegible]<00 00 00 00 00 00 00 00 00 00 00 00 00  : ...B............
```

```
BOUCLE  CLC
        LDA  #$80
        ADC  LB
        STA  LB
        LDA  #$00
        ADC  HB
        STA  HB
        DEX
        BNE  BOUCLE
END     RTS
*************
**.........**
**..SOUND..**
**.........**
*************
SOUND   LDX  #$FF
SOUND1  LDA  #$58
SOUND2  LDY  #$1B
SOUND3  DEY
        BNE  SOUND3
        BIT  SPKR
        TAY
SOUND4  DEY
        BNE  SOUND4
        SBC  #$01
        BEQ  SOUND1
        BIT  SPKR
        DEX
        BNE  SOUND2
        RTS
*********************
**................**
**..STROUT.&.COUT..**
**................**
*********************
STROUT40 STA  LB.
         STY  HB.
         LDY  #$00
STR.40   TYA
         PHA
         LDA  (LB.),Y
         BNE  STR.40.
         PLA
         RTS
STR.40.  JSR  COUT.40
         PLA
         TAY
         INY
         JMP  STR.40
COUT.40  CMP  #$8D
         BEQ  RET40
         STA  CAR
         JSR  COL.LIG
         INC  COL
         INC  COL
         RTS
RET40    INC  LIG
         LDA  #$01
         STA  COL
         RTS
STROUT80 STA  LB.
         STY  HB.
         LDY  #$00
STR.80   TYA
         PHA
         LDA  (LB.),Y
         BNE  STR.80.
         PLA
         RTS
STR.80.  JSR  COUT.80
         PLA
         TAY
         INY
         JMP  STR.80
COUT.80  CMP  #$8D
         BEQ  RET80
         STA  CAR
         JSR  COL.LIG
         INC  COL
```

```
6E48- D6 C5 BA A0 31 AF B2 A0
6E50- A0 CF C3 D4 C5 D4 BA A0
6E58- A4 A0 A0 A0 A0 A0 A0 A0
6E60- A0 A0 5F DE D8 A0 BA A0
6E68- D3 F7 E9 F4 E3 E8 E5 F2
6E70- A0 C3 E1 F2 E1 E3 F4 E5
6E78- F2 E5 F3 A0 A0 5F A0 A0
6E80- 5A 8D 00 A0 DF DF DF DF
6E88- DF DF DF DF DF DF DF DF
6E90- DF DF DF DF DF DF DF DF
6E98- DF DF DF DF A0 8D 00 5A
6EA0- A0 A0 A0 A0 A0 A0 A0 A0
6EA8- A0 A0 A0 A0 A0 A0 A0 A0
6EB0- A0 A0 A0 A0 A0 A0 A0 A0
6EB8- 5F 8D 00 5A A0 BE BE BE
6EC0- 05 72 72 65 75 72 20 04
6EC8- 69 73 71 75 65 74 74 65
6ED0- BC BC BC A0 5F 8D 00 5A
6ED8- A0 A0 A0 A0 A0 C1 D0 D0
6EE0- D5 D9 C5 DA A0 D3 D5 D2
6EE8- A0 00 4D A0 A0 A0 A0 A0
6EF0- A0 5F 8D 00 D2 C5 D4 D5
6EF8- D2 CE A0 5F 8D 00 A0 4C
6F00- 4C 4C 4C 4C 4C 4C 4C 4C
6F08- 4C 4C 4C 4C 4C 4C 4C 4C
6F10- 4C 4C 4C 4C 4C 4C 4C A0
6F18- 00 C9 CC A0 C6 C1 D5 D4
6F20- A0 D5 CE C5 A0 C3 C1 D2
6F28- D4 C5 A0 B8 B0 A0 C3 CF
6F30- CC CF CE CE C5 D3 00 BF
6F38- 19 04 00 00 00 00 00 00
6F40- 00 00 00 00 00
```

```
        RTS
RET80   INC  LIG
        LDA  #$00
        STA  COL
        RTS
****************
**............**
**..MESSAGES..**
**............**
****************
PRES.0  DS   4,$8D
        DS   8," "
        DS   23,"_"
        DFB  $8D
        DS   7," "
        DFB  $5A
        DS   23," "
        DFB  $5F,$8D
        DS   7," "
        DFB  $5A
        DS   6," "
        ASC  "R.S.P.D.:"
        DS   8," "
        DFB  $5F,$8D,$00
PRES.1  DS   7," "
        DFB  $5A
        DS   6," "
        DS   9,'L'
        DS   8," "
        DFB  $5F,$8D
        DS   7," "
        DFB  $5A
        ASC  " (Read & Save Pro
             Dos) "
        DFB  $5F,$8D
        DS   7," "
        DFB  $5A
        DS   23," "
        DFB  $5F,$8D,$00
PRES.2  DS   7," "
        DFB  $5A
        ASC  " Utilitaire de Le
             cture "
        DFB  $5F,$8D
        DS   7," "
        DFB  $5A
        ASC  " et d'ecriture di
             recte "
        DFB  $5F,$8D
        DS   7," "
        DFB  $5A
        ASC  " sur Disquettes P
             roDOS "
        DFB  $5F,$8D
        DS   7," "
        DFB  $5A
        DS   23," "
        DFB  $5F,$8D,$00
PRES.3  DS   7," "
        DFB  $5A
        ASC  "   par Marc ROGLI
             ANO   "
        DFB  $5F,$8D
        DS   7," "
        DFB  $5A
        DS   23," "
        DFB  $5F,$8D
        DS   8," "
        DS   23,$4C
        DS   4,$8D
        DS   15," "
        ASC  "Appuyez sur "
        DFB  $00
PRES.4.1 DFB $4D,$A0,$4E,$8D
        DS   15," "
        DS   13,$4C
        DFB  $8D,$00
PRES.4.2 ASC "RETURN "
        DFB  $20,$8D
        DS   15," "
        DS   18,$4C
        DFB  $8D,$00
AAA0    ASC  "R.S.P.D.: Utilita
             ire de Lecture et
             d'Ecriture sur dis
             ks ProDOS par Marc
             ROGLIANO"
        DFB  $8D
        ASC  "  Commandes:"
        DFB  $5A,$A0,$00
AAA0.1  DFB  $48,$A0,$55,$A0
        DFB  $4B,$A0,$4A,$00
AAA0.2  ASC  "fleches"
        DFB  $00
AAA1    ASC  " : Deplacements s
             ur l'ecran "
        DFB  $5F
        ASC  "^A : Entrer carac
             teres"
        DS   4," "
        DFB  $5F,$8D
        DS   2," "
        DS   9,$4C
        DFB  $A0,$5A,$A0,$00
AAA1.1  DFB  $40,$A0,$48
        DFB  $A0,$55
        ASC  " : Deplacements s
             ur Disquette "
        DFB  $00
AAA1.2  ASC  "Pomme fermee <- -
             > : '' sur Disque
             "
        DFB  $00
AAA2    DFB  $5F
        ASC  "^B : Entrer un Bl
             ock"
        DS   6," "
        DFB  $5F,$A0
        DFB  $A0,$51,$8D
        ASC  "   (ESC:    "
        DFB  $5A,$A0,$00
AAA2.1  DFB  $41,$A0
        DFB  $48,$A0,$55
        ASC  " : Deplacements (
             *10)"
        DS   9," "
        DFB  $00
AAA2.2  ASC  "Pomme ouverte <-
             -> : Depl. x 10
             "
        DFB  $00
        DFB  $5F
AAA3    ASC  "^C : Changer de L
             ecteur"
        DS   3," "
        DFB  $5F,$A0,$A0
        DFB  $5A,$8D
        ASC  "   Quitter) "
        DFB  $5A
        ASC  "  ESPACE : Change
             r de Page"
        DS   10," "
        DFB  $5F
        ASC  "^S : Sauver sur D
             isquette "
        DFB  $5F,$A0,$A0,$5A,
             $8D
        ASC  " BLOCK:"
        DS   5," "
        ASC  "($   )  DRIVE: "
        DFB  $31
        ASC  "/2  OCTET: $"
        DS   9," "
        DFB  $5F
        ASC  "^X : Switcher Car
             acteres  "
        DFB  $5F,$A0,$A0
        DFB  $5A,$8D,$00
ERROR.0 ASC  " "
        DS   24,"_"
        ASC  " "
        DFB  $8D,$00
ERROR.1 DFB  $5A
        DS   24," "
        DFB  $5F,$8D,$00
ERROR.2 DFB  $5A,$A0
        ASC  ">>>"
        DFB  $05,$72,$72,$65
        DFB  $75,$72,$20,$04
        DFB  $69,$73,$71,$75
        DFB  $65,$74,$74,$65
        ASC  "<<<"
        DFB  $A0,$5F,$8D,$00
ERROR.3 DFB  $5A
        DS   5," "
        ASC  "APPUYEZ SUR "
        DFB  $00
ERROR4.1 DFB $4D
        DS   6," "
        DFB  $5F,$8D,$00
ERROR4.2 ASC "RETURN "
        DFB  $5F,$8D,$00
ERROR.5 ASC  " "
        DS   24,$4C
        ASC  " "
        DFB  $00
MESSERR ASC  "IL FAUT UNE CARTE
             80 COLONNES"
        DFB  $00
        DFB  $BF,$19,$04
SAVE    DS   4
TABLE   DS   7
TABLE1  EQU  *
```

Roland Jost

Miroirs en HGR

Janus

Voici un programme qui permet de manipuler des images graphiques en donnant des effets de miroir. Il est aussi le prétexte à une étude de l'organisation des pages graphiques de Apple. Il permet aussi d'illustrer le diffici e compromis entre gain de temps et gain de place.

La page graphique de l'Apple

La haute résolution de l'Apple est de 280 x 192 = 53760 points. Or une page HGR ne mobilise que 8192 octets. Ceci est obtenu en stockant l'état de 7 points consécutifs d'une ligne dans les bits 0 à 6 d'un octet, le bit 7 contenant le mode couleur. On réduit ainsi de 8 fois la taille mémoire nécessaire pour la page graphique. L'inconvénient est que cela ne facilite pas l'adressage point par point. Le problème serait assez simple à résoudre, mais l'écran HGR est divisé en 3 zones de 8 groupes de 8 lignes. En mémoire, les 192 lignes de 40 octets ne se suivent donc pas de facon séquentielle (cf les adresses des lignes dans POM'S n°1). Ces adresses peuvent se calculer, par exemple par la formule utilisée dans Édigraph (POM'S n°21).

Mais même en assembleur, ce calcul prend du temps, surtout s'il doit être répété de nombreuses fois. Une méthode plus rapide, utilisée dans de nombreux jeux est l'utilisation d'une table d'adresses, en fait de deux tables, la première contenant l'octet bas, la seconde l'octet haut. Cette méthode a l'inconvénient de prendre beaucoup de place (384 octets). On peut trouver un compromis entre ces deux méthodes en utilisant la régularité de la table pour huit lignes successives. La table est alors huit fois moins importante (seulement 48 octets). Il faudrait également une table des octets hauts différente pour la page HGR et pour la page HGR2. Une table unique suffit, à condition de rajouter la valeur contenue dans $E6 ($20 pour la page HGR, $40 pour la page HGR2). JANUS sera donc actif sur la page en cours, sans qu'il soit nécessaire de spécifier son numéro. Au prix d'un léger ralentissement du programme (environ un centième de seconde), on parvient donc à réduire l'encombrement de la table des adresses de 476 octets à seulement 48 octets.

Miroirs

Obtenir l'image dans un miroir d'une page graphique est facile quand on a calculé les adresses des lignes.

Pour un miroir horizontal, il suffit de permuter la première ligne avec la dernière, la deuxième avec l'avant-dernière, etc.

Pour un miroir vertical, le principe est de permuter la colonne 0 (192 octets) avec la colonne 39, la colonne 1 avec la colonne 38, etc. Mais il faut de plus prendre l'image dans un miroir des bits 0 à 6 de chaque octet sans toucher au bit 7, c'est-à-dire passer par exemple de 01101000 à 00001011.

Là encore ce travail peut être effectué soit par un sous-programme comme celui qui est

Comment faire ?

Saisir puis sauvegarder la récapitulation 'JANUS'.
Dans un programme Basic, il faut, pour bénéficier des effets de miroirs, charger la routine par BLOAD JANUS (se reporter au programme de démonstration).

Les possibilités de la routine sont les suivantes :

```
CALL   36864,H,0   Retournement horizontal
CALL   36864,V,0   gauche <=> droite
CALL   36864,H,1   Moitié supérieure réfléchie dans moitié inférieure
CALL   36864,V,1   Moitié gauche réfléchie dans moitié droite
```

De plus, on dispose des combinaisons suivantes :

```
CALL   36864,H,0   :  CALL   36864,H,1
                      Moitié gauche dans partie droite
CALL   36864,V,0   :  CALL   36864,H,0
                      Rotation 180°, combinaison des 2 symétries
```

présenté ci-dessous, soit à l'aide d'une table.

```
ROLL   STA $08    ; sauve l'octet à
                  inverser
       ROL A      ; le bit 7 est mis
                  dans la retenue
       ROL A      ; décale A à gauche et
                  retenue dans bit 0
       LDY #$07   ; il faut le faire 7
                  fois
ROLL1  LSR $08    ; le bit 0 est mis
                  dans la retenue
       ROL A      ; retenue dans bit 0
                  et décale à gauche
       DEY
       BNE ROLL1
       RTS
```

Avec la première solution l'inversion se fait en plus d'une seconde ; avec la seconde le programme va trois fois plus vite. Ceci s'explique facilement par le fait que chacun des octets doit être permuté, soit 8192 permutations. Le moindre gain de vitesse à ce niveau est donc plus qu'appréciable. C'est pourquoi nous choisirons cette dernière méthode.

JANUS permet encore d'obtenir dans la moitié inférieure de l'écran, le reflet de la partie supérieure. De même pour les parties gauche et droite. Il s'agit dans les deux cas d'arrêter la recopie au milieu de l'écran.

Récapitulation 'JANUS'

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE JANUS, A$9000, L$200

```
9000- 20 BE DE 85 EF 20 B1 00
9008- 20 F5 E6 86 1E A2 00 86
9010- EB 86 ED A2 27 86 EC A2
9018- BF 86 EE A5 EF C9 48 F0
9020- 05 C9 56 F0 3E 60 E6 EC
9028- A6 ED 86 06 A6 EE 86 07
9030- A6 06 20 AD 90 85 18 84
9038- 19 A6 07 20 AD 90 85 1A
9040- 84 1B A4 EB B1 18 A6 1E
9048- D0 06 48 B1 1A 91 18 68
9050- 91 1A C8 C4 EC D0 ED E6
9058- 06 C6 07 38 A5 07 E5 06
9060- B0 CE 60 E6 EE A6 ED 20
9068- AD 90 85 18 84 19 A4 EB
9070- 84 06 A4 EC 84 07 A4 06
9078- B1 18 A8 B9 00 91 A4 1E
9080- D0 0C 48 A4 07 B1 18 A8
9088- B9 00 91 85 1C 68 A4 07
9090- 91 18 A4 1E D0 06 A5 1C
9098- A4 06 91 18 E6 06 C6 07
90A0- 38 A5 07 E5 06 B0 CF E8
90A8- E4 EE D0 BB 60 8A 4A 4A
90B0- 4A A8 B9 C4 90 48 8A 29
90B8- 07 0A 0A 18 79 DC 90 65
90C0- E6 A8 68 60 00 80 00 80
90C8- 00 80 00 80 28 A8 28 A8
90D0- 28 A8 28 A8 50 D0 50 D0
90D8- 50 D0 50 D0 00 00 01 01
90E0- 02 02 03 03 00 00 01 01
90E8- 02 02 03 03 00 00 01 01
90F0- 02 02 03 03 00 00 00 00
90F8- 00 00 00 00 00 00 00 00
9100- 00 40 20 60 10 50 30 70
9108- 08 48 28 68 18 58 38 78
9110- 04 44 24 64 14 54 34 74
9118- 0C 4C 2C 6C 1C 5C 3C 7C
9120- 02 42 22 62 12 52 32 72
9128- 0A 4A 2A 6A 1A 5A 3A 7A
9130- 06 46 26 66 16 56 36 76
9138- 0E 4E 2E 6E 1E 5E 3E 7E
9140- 01 41 21 61 11 51 31 71
9148- 09 49 29 69 19 59 39 79
9150- 05 45 25 65 15 55 35 75
9158- 0D 4D 2D 6D 1D 5D 3D 7D
9160- 03 43 23 63 13 53 33 73
9168- 0B 4B 2B 6B 1B 5B 3B 7B
9170- 07 47 27 67 17 57 37 77
9178- 0F 4F 2F 6F 1F 5F 3F 7F
9180- 80 C0 A0 E0 90 D0 B0 F0
9188- 88 C8 A8 E8 98 D8 B8 F8
9190- 84 C4 A4 E4 94 D4 B4 F4
9198- 8C CC AC EC 9C DC BC FC
91A0- 82 C2 A2 E2 92 D2 B2 F2
91A8- 8A CA AA EA 9A DA BA FA
91B0- 86 C6 A6 E6 96 D6 B6 F6
91B8- 8E CE AE EE 9E DE BE FE
91C0- 81 C1 A1 E1 91 D1 B1 F1
91C8- 89 C9 A9 E9 99 D9 B9 F9
91D0- 85 C5 A5 E5 95 D5 B5 F5
91D8- 8D CD AD ED 9D DD BD FD
91E0- 83 C3 A3 E3 93 D3 B3 F3
91E8- 8B CB AB EB 9B DB BB FB
91F0- 87 C7 A7 E7 97 D7 B7 F7
91F8- 8F CF AF EF 9F DF BF FF
```

Programme 'JANUS.DEMO'

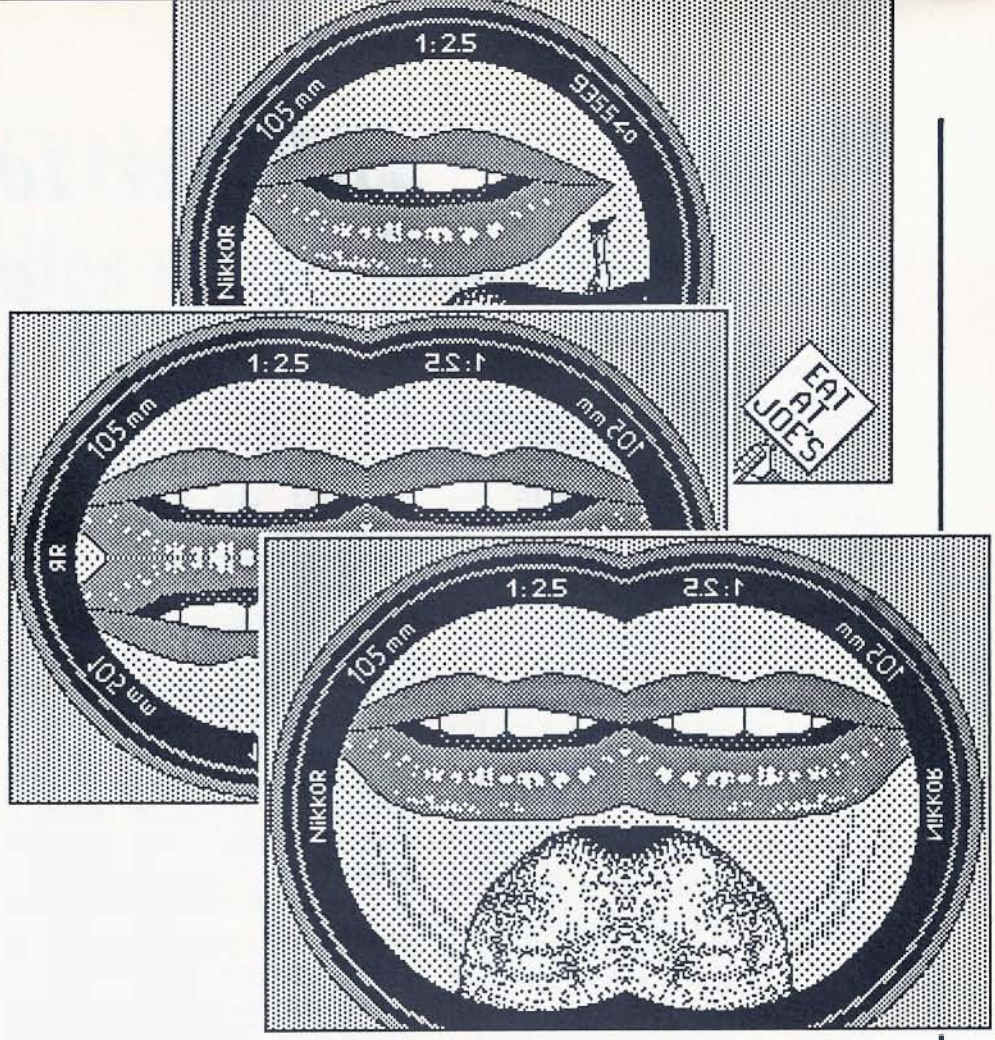
```
10  REM  DEMO.JANUS
12  REM  R.JOST 1985
15  PRINT  CHR$ (4)"BLOAD JANUS"
20  TEXT : HOME
30 A = 36864
40 N$ = "IMAGE"
50  HGR : POKE  - 16302,0
60  GOSUB 600
65  GOSUB 500
70  CALL A,H,0: GOSUB 500
80  CALL A,H,0: GOSUB 500
90  CALL A,V,0: GOSUB 500
100  CALL A,V,0: GOSUB 500
110  CALL A,V,1: GOSUB 500
120  GOSUB 600
124  CALL A,H,0: CALL A,H,1
126  GOSUB 500: GOSUB 600
130  CALL A,V,0: GOSUB 500
140  CALL A,V,1: GOSUB 500
144  CALL A,H,0: CALL A,H,1
146  GOSUB 500
150  GOSUB 600
170  GOSUB 500
180  TEXT : END
500  FOR T = 1 TO 2000: NEXT :ZZ =
     PEEK ( - 16336) +  PEEK ( - 163
     36): RETURN
600  PRINT  CHR$ (4)"BLOAD "N$",A$2
     000": RETURN
```

Source 'JANUS.S'

Assembleur MERLIN

```
1 ******************************
2 *                            *
3 *            JANUS           *
4 *                            *
5 ******************************
6 *
7 * R.JOST
8 * Assembleur MERLIN
9 *
10 * REFLET D'UNE PAGE HGR
11 *
12 * Syntaxe : CALL 36864,MODE,OPT
13 *
14 *  MODE=H:MIROIR HORIZONTAL
15 *       =V:MIROIR VERTICAL
16 *OPTION=0:IMAGE DE TOUT L'ECRAN
17 *       =1:MOITIE ECRAN REFLECHI
           E DANS L'AUTRE
18 *SUPERIEURE -> INFERIEURE
19 *GAUCHE -> DROITE
20 *
21 ******************************
22 *
23 *
24 Z1        EQU  $06
25 Z2        EQU  $07
26 DEPART    EQU  $18
27 ARRIVEE   EQU  $1A
28 TAMPON    EQU  $1C
29 OPTION    EQU  $1E
30 PAGE      EQU  $E6
31 X1        EQU  $EB
32 X2        EQU  $EC
33 Y1        EQU  $ED
34 Y2        EQU  $EE
```

Ayez sur votre disquette une image HGR nommée "IMAGE"




```
35 MODE      EQU  $EF
36
37 CHRGET    EQU  $00B1
38 CHKCOM    EQU  $DEBE
39 GETBYTC   EQU  $E6F5
40 *
41 *
42           ORG  $9000
43 *
44 *
45 * SAISIE DES PARAMETRES
46 *
47           JSR  CHKCOM
48           STA  MODE
49           JSR  CHRGET
50           JSR  GETBYTC
51           STX  OPTION
52           LDX  #0
53           STX  X1
54           STX  Y1
55           LDX  #39
56           STX  X2
57           LDX  #191
58           STX  Y2
59           LDA  MODE
60           CMP  #$48
61           BEQ  I1
62           CMP  #$56
63           BEQ  I2
64           RTS
65 *
66 *         MIROIR HORIZONTAL
67 *
68 I1        INC  X2
69           LDX  Y1
70           STX  Z1
71           LDX  Y2
72           STX  Z2
73 BOU1      LDX  Z1
74           JSR  CALCADR
75           STA  DEPART
76           STY  DEPART+1
77
78           LDX  Z2
79           JSR  CALCADR
80           STA  ARRIVEE
81           STY  ARRIVEE+1
82
83           LDY  X1
84 BOU2      LDA  (DEPART),Y
85 *
86           LDX  OPTION
87           BNE  SUIT1
88           PHA
89           LDA  (ARRIVEE),Y
90           STA  (DEPART),Y
91           PLA
92 SUIT1     STA  (ARRIVEE),Y
93           INY
94           CPY  X2
95           BNE  BOU2
96           INC  Z1
97           DEC  Z2
98           SEC
99           LDA  Z2
100          SBC  Z1
101          BCS  BOU1
102          RTS
103 *
104 * MIROIR VERTICAL
105 *
106 I2       INC  Y2
107          LDX  Y1
108 BO1      JSR  CALCADR
109          STA  DEPART
110          STY  DEPART+1
111
112          LDY  X1
113          STY  Z1
114          LDY  X2
115          STY  Z2
116 BO2      LDY  Z1
117          LDA  (DEPART),Y
118          TAY
119          LDA  PERMUT,Y
120          LDY  OPTION
121          BNE  SUIT2
122          PHA
123          LDY  Z2
124          LDA  (DEPART),Y
125          TAY
126          LDA  PERMUT,Y
127          STA  TAMPON
128          PLA
129 SUIT2    LDY  Z2
130          STA  (DEPART),Y
131          LDY  OPTION
132          BNE  SUIT3
133          LDA  TAMPON
134          LDY  Z1
135          STA  (DEPART),Y
136 SUIT3    INC  Z1
137          DEC  Z2
138          SEC
139          LDA  Z2
140          SBC  Z1
141          BCS  BO2
142          INX
143          CPX  Y2
144          BNE  BO1
145          RTS
146 *
147 *  Calcul l'adresse d'une ligne
148 *
149 CALCADR  TXA
150          LSR
151          LSR
152          LSR
153          TAY
154          LDA  TABIL,Y
155          PHA
156          TXA
157          AND  #%00000111
158          ASL
159          ASL
160          CLC
161          ADC  TABIH,Y
162          ADC  PAGE
163          TAY
164          PLA
165          RTS
166 *
167 * TABLES DES ADRESSES HGR
168 *        OCTET BAS
169 *
170 TABIL    DFB  $00,$80,$00,$80,$
             00,$80,$00,$80
171          DFB  $28,$A8,$28,$A8,$
             28,$A8,$28,$A8
172          DFB  $50,$D0,$50,$D0,$
             50,$D0,$50,$D0
173 *
174 * TABLES DES ADRESSES HGR
175 *        OCTET HAUT
176 * RAJOUTER $20 POUR PAGE 1
177 * OU $40 POUR PAGE 2
178 *
179 TABIH    DFB  $00,$00,$01,$01,$
```

```
                 02,$02,$03,$03
180        DFB   $00,$00,$01,$01,$
                 02,$02,$03,$03
181        DFB   $00,$00,$01,$01,$
                 02,$02,$03,$03
182 *
183 * TABLE D'INVERSION
184 *
185        DS    $9100-*
186 PERMUT DFB   $00,$40,$20,$60,$
                 10,$50,$30,$70
187        DFB   $08,$48,$28,$68,$
                 18,$58,$38,$78
188        DFB   $04,$44,$24,$64,$
                 14,$54,$34,$74
189        DFB   $0C,$4C,$2C,$6C,$
                 1C,$5C,$3C,$7C
190        DFB   $02,$42,$22,$62,$
                 12,$52,$32,$72
191        DFB   $0A,$4A,$2A,$6A,$
                 1A,$5A,$3A,$7A
192        DFB   $06,$46,$26,$66,$
                 16,$56,$36,$76
193        DFB   $0E,$4E,$2E,$6E,$
                 1E,$5E,$3E,$7E
194        DFB   $01,$41,$21,$61,$
                 11,$51,$31,$71
195        DFB   $09,$49,$29,$69,$
                 19,$59,$39,$79
196        DFB   $05,$45,$25,$65,$
                 15,$55,$35,$75
197        DFB   $0D,$4D,$2D,$6D,$
                 1D,$5D,$3D,$7D
198        DFB   $03,$43,$23,$63,$
                 13,$53,$33,$73
199        DFB   $0B,$4B,$2B,$6B,$
                 1B,$5B,$3B,$7B
200        DFB   $07,$47,$27,$67,$
                 17,$57,$37,$77
201        DFB   $0F,$4F,$2F,$6F,$
                 1F,$5F,$3F,$7F
202
203        DFB   $80,$C0,$A0,$E0,$
                 90,$D0,$B0,$F0
204        DFB   $88,$C8,$A8,$E8,$
                 98,$D8,$B8,$F8
205        DFB   $84,$C4,$A4,$E4,$
                 94,$D4,$B4,$F4
206        DFB   $8C,$CC,$AC,$EC,$
                 9C,$DC,$BC,$FC
207        DFB   $82,$C2,$A2,$E2,$
                 92,$D2,$B2,$F2
208        DFB   $8A,$CA,$AA,$EA,$
                 9A,$DA,$BA,$FA
209        DFB   $86,$C6,$A6,$E6,$
                 96,$D6,$B6,$F6
210        DFB   $8E,$CE,$AE,$EE,$
                 9E,$DE,$BE,$FE
211        DFB   $81,$C1,$A1,$E1,$
                 91,$D1,$B1,$F1
212        DFB   $89,$C9,$A9,$E9,$
                 99,$D9,$B9,$F9
213        DFB   $85,$C5,$A5,$E5,$
                 95,$D5,$B5,$F5
214        DFB   $8D,$CD,$AD,$ED,$
                 9D,$DD,$BD,$FD
215        DFB   $83,$C3,$A3,$E3,$
                 93,$D3,$B3,$F3
216        DFB   $8B,$CB,$AB,$EB,$
                 9B,$DB,$BB,$FB
217        DFB   $87,$C7,$A7,$E7,$
                 97,$D7,$B7,$F7
218        DFB   $8F,$CF,$AF,$EF,$
                 9F,$DF,$BF,$FF
```

Inhiber le Reset en ProDOS

Bruno Fénart

La méthode pour inhiber le CTRL-RESET est bien connue en DOS 3.3 (voir "clef pour l'Apple //e" de N. Bréaud-Pouliquen aux Éditions du PSI, ou "All About Dos" de Call-Apple). En ProDOS, c'est plus compliqué car il faut revectoriser soi-même les vecteurs d'entrée/sortie.

Quelle que soit la solution retenue, la routine de revectorisation devra réinitialiser la pile pour éviter le 'Out Of Memory Error' après quelques RESETs. La routine suivante pourrait être adoptée :

```
3F2:00 03 A6                    ;Revectorisation du reset
300:A2 03        LDX  #$03
302:BD 34 BE     LDA  $BE34,X   ;vecteurs d'E/S ProDOS
305:95 36        STA  $36,X     ;vecteurs d'E/S
307:CA           DEX
308:10 F8        BPL  $302
30A:A2 FF        LDX  #$FF      ;Réinitialise la pile
30C:9A           TXS
30D:20 98 D8     JSR  $D898     ;CONT
310:4C D2 D7     JMP  $D7D2     ;NEWSTART
```

Cette solution peut entraîner des erreurs et elle n'est pas utilisable si l'affichage du programme est en 80 colonnes. C'est pourquoi on peut avoir intérêt à utiliser une autre solution qui consisterait non pas à retourner à la ligne où a eu lieu le CTRL-RESET, mais comme pour un traitement d'erreur, à effectuer un branchement vers une ligne donnée (ici vers la ligne 20). Si cette ligne n'existe pas on obtient "?UNDEF'D STATEMENT ERROR" comme pour un GOTO sur une ligne non définie :

```
300:A2 03        LDX  #$03
302:BD 34 BE     LDA  $BE34,X   ;vecteurs d'E/S ProDOS
305:95 36        STA  $36,X     ;vecteurs d'E/S
307:CA           DEX
308:10 F8        BPL  $302
30A:A2 FF        LDX  #$FF      ;Réinitialise la pile
30C:9A           TXS
30D:A9 14        LDA  #$14      ;Ligne n°20
30F:85 50        STA  $50       ;LINNUM
311:A9 00        LDA  #$00
313:85 51        STA  $51       ;LINNUM+1
315:20 41 D9     JSR  $D941     ;GOTO+3
318:4C D2 D7     JMP  $D7D2     ;NEWSTART
```

Le programme Basic devra alors avoir la forme suivante :

```
10 GOTO 1000
20 GOTO 1500
1000 REM Début du programme
1500 REM Prise en charge après le ctrl-reset
```

Claude *Vivier*

'COMPDISK' ou 'sont-elles vraies jumelles ?'

Qui d'entre nous n'a jamais eu entre les mains des versions "peut-être" identiques d'un jeu ou d'un quelconque logiciel situé sur des disquettes non structurées en fichiers et/ou protégées. Dans cette situation -pas quotidienne il est vrai- il est naturel de chercher à comparer de telles disquettes. Mais, si les programmes de comparaison de fichiers ne manquent pas sur le marché, on ne peut en dire autant des comparateurs de disquettes vues globalement. Une solution consiste à utiliser le programme "Zap" du Bag of Tricks en construisant une grosse macro-commande, encore faut-il que les disquettes se laissent faire !...

Voici donc un programme en assembleur qui permet de comparer toutes les disquettes seize secteurs de format standard ou légèrement "hors-normes".

Le principe

Il a paru intéressant de travailler au niveau de la piste complète transférée en mémoire plutôt que d'accéder au disque secteur par secteur. Bien sûr, en dernier ressort il faut arriver à la notion de secteur mais, d'une part, il y a gain de temps du fait du nombre limité d'accès disques, et ensuite nous sommes moins tributaires des gadgets des protecteurs de logiciels, les contrôles ayant été volontairement limités au strict nécessaire. Il n'est fait aucune référence à la notion de fichier.

Utilisation

Le programme doit être situé sur une disquette DOS 3.3 ou apparentée et nécessite deux lecteurs. L'affichage 80 colonnes et les minuscules ont été écartés de façon à s'adapter à toutes les configurations.

Il se lance par :

```
BRUN COMPDISK.
```

Un premier écran demande

- d'installer les deux disquettes sur les deux lecteurs ;
- d'indiquer le type des disquettes testées (Dos, Pascal, Prodos, CP/M, ou Inconnu si on l'ignore) au moyen d'un chiffre de 1 à 4 ; ceci afin de pouvoir afficher un numéro de secteur correct en cas de différences.

Il faut rappeler, en effet, que pour chaque système d'exploitation il existe une table de corrélation entre le numéro de secteur physique (celui qui est écrit sur le disque) et le numéro de secteur logique (celui par lequel l'utilisateur accède à ce secteur). Si l'on a entré l'option 4 (système inconnu), c'est le numéro de secteur physique qui sera affiché, bien entendu.

Comment faire ?

Après avoir saisi et sauvegardé la récapitulation COMPDISK, sur une disquette DOS 3.3, pour comparer deux disquettes, faire :

```
BRUN COMPDISK
```

puis indiquer le type de DOS s'il est connu. Valider alors par ESC ou RETURN selon que la disquette est protégée ou non. Dans le premier cas, vous devrez indiquer le prologue obtenu à l'aide d'un utilitaire du type TRAX. Le défilement des secteurs qui diffèrent est contrôlé par ESPACE.

Le programme demande ensuite s'il s'agit d'une disquette *normale* (en gros, copiable par COPYA) ou "protégée". Dans le premier cas, <Return> fait partir la comparaison. Dans le deuxième cas, <ESC> amène un deuxième écran qui explique que les disques standard sont formatés avec, pour chaque secteur le prologue d'adresse D5AA96, et le prologue de Data D5AAAD (dans le programme, le nibble AD n'est pas exigé). Certaines disquettes protégées répondent à ce schéma, d'autres non. Il existe plusieurs programmes qui montrent ces prologues (le Trax du Bag of Tricks par exemple).

Il est alors demandé ou bien de frapper <Return> pour accepter les prologues proposés, ou bien de saisir les prologues des disques à comparer. Finalement un <Return> de confirmation fera partir le programme.

La progression du programme est suivie par affichage des pistes testées, et, en cas de différence, des numéros de secteur, de la première position différente dans le secteur, et pour les disquettes Prodos, Pascal et CP/M le numéro de bloc et de partie de bloc. En fin de parcours, on pourra comparer un autre couple de disques, sans recharger le programme.

Analyse du programme

Lignes 103-225

- Patch du programme pour valeurs standard des prologues.
- Affichage des choix.
- Saisie des paramètres (système et nibbles de reconnaissance).
- Patch du programme avec ces différents paramètres.

Lignes 226-250

• calage du bras de lecteur 2 sur la piste 0 (routine de la PROM du contrôleur) ; opération indispensable pour éviter de gros déboires dans la recherche des pistes.

Lignes 253-307

• boucle générale de traitement. Chaque couple de pistes de 0 à 34 est transféré en mémoire ($2000 pour I et $4000 pour II) pour être comparé. A la fin, retour possible au début ou sortie vers le warmstart du Dos.

Lignes 311-371

• routine de Dump mémoire d'une piste.

• Alimentation de la "parameter list" qui contrôle la procédure RWTS ; on met 0 dans le paramètre de "commande" (Search)

• Appel à RWTS, qui amène le bras du lecteur sur la piste choisie sans manipuler encore d'informations. Pour cette étape, il n'y a pas eu de référence au numéro de piste codé dans le champ d'adresse des secteurs de cette piste (code fréquemment altéré sur les disques protégés).

• Lecture des nibbles de la piste vers la mémoire. A noter que les instructions 362-369 sont très critiques. Pour qu'il n'y ait pas de perte d'information, un petit calcul montre que le délai entre deux lectures de nibbles ne doit pas excéder 31 microsecondes, on y arrive tout juste. L'ensemble de cette routine de dump est valable pour les 2 pistes I et II. $2000 nibbles sont ainsi transférés, soit très largement une fois la piste.

Lignes 375-423

• Boucle générale de comparaison des 16 secteurs de la piste.

• Recherche du premier secteur disponible de la piste 1, normalement annoncé par D5AA96, avec sauvegarde, en passant, des nibbles donnant le numéro de secteur (physique, bien sûr).

• Recherche du secteur homologue sur piste II (voir plus loin).

• Comparaison des 2 secteurs (voir plus loin).

• On continue sur le secteur suivant de la piste I jusqu'au 16ème. A chaque fois que l'on explore la piste II à la recherche du secteur frère, on repart toujours du début de la piste, alors qu'on serait tenté, une fois qu'un couple de secteurs est "accroché", de poursuivre l'exploration séquentiellement sur les 2 pistes, quitte à dumper au moins 2 fois la piste II. Ce choix a été fait afin d'éviter de déclarer différentes des disquettes formatées avec des "entrelacements" différents, alors qu'elles sont identiques quant au contenu. Certains copieurs ne respectent pas l'entrelacement de l'original (COPYA par exemple qui commence une copie par un INIT standard). La perte de temps qui résulte de cette manière de faire est d'ailleurs tout à fait négligeable.

Lignes 427-478

• Procédure de recherche du secteur homologue II. Dès qu'un secteur est trouvé, son numéro est testé par rapport au numéro sauvegardé du secteur I. Au début de cette routine, la frappe de n'importe quelle touche suspend le traitement. Si on ne trouve pas sur toute la piste (de même d'ailleurs pour la piste I), les nibbles de prologue, tels qu'ils ont été définis en début de programme, la piste est déclarée illisible et affichée comme telle (lignes 684 à 728). Dès que les pointeurs de balayage des 2 pistes sont en début de la partie Data du secteur, la comparaison peut commencer. On sauvegarde à ce moment les valeurs des pointeurs (lignes 474-478).

Lignes 482-493

• Routine de comparaison des 342 nibbles du secteur. Dès qu'il y a discordance, la comparaison est arretée et il y a branchement sur...

Lignes 504-656

• Gestion de la première différence trouvée sur le secteur. Il s'agit de trouver la position en cause non pas en termes de nibbles mais d'octets "visibles" tels qu'on les connait habituellement. Il semble, après longue réflexion, que la meilleure solution pour avoir, à coup sûr, la première divergence est de recourir à la dénibbilisation classique de la routine RWTS. Le secteur I est complètement décodé en $7100 (buffer secondaire en $7000). (Donc si on frappe sur Reset en cours de traitement, un coup d'œil en $7100 donnera le dernier secteur trouvé différent sous forme décodée). On fait de même pour le secteur II en arrêtant la procédure dès qu'il y a discordance (buffers en $8100 et $8000). La position trouvée est alors mise en réserve (ligne 552). Ensuite vient la routine d'affichage des résultats trouvés (piste, secteur logique, position), avec, s'il s'agit de la première différence trouvée, affichage d'un en-tête de présentation (lignes 553 à 574).

• Un petit calcul (lignes 612-656) donne le bloc en cause s'il y a lieu. Une petite remarque au sujet des instructions 545 et 550 : il se peut que l'on trouve une différence à l'examen des nibbles et que celle-ci disparaisse après dénibbilisation. La raison en est que les nibbles de rang $54,55 ne sont que partiellement utilisés dans la dénibbilisation; ceci explique que le décodage du secteur est fait avant d'ameuter les populations.

Lignes 660-905

Routines diverses de sorties de message, saisie des prologues, conversion hexa, patches etc...

Performances

Durée du traitement :
Deux disquettes identiques : 2 mn 05 s
Disquettes complètement différentes : 2 mn 20 s
Temps consacré à la recherche des pistes : 1 mn 30 s

Attention : Ce programme ne fonctionnera pas ou très mal avec des disquettes 13 secteurs. Il faut se méfier de certaines disquettes déguisées en 16 secteurs mais qui en fait sont des 13 secteurs. Elles 'bootent' normalement, mais dès

le premier étage du boot il y a transformation du Dos en l'autre format.

Pour ceux qui en auraient encore douté, ce petit travail n'aurait pas pu etre fait sans références permanentes au livre de Don Worth et Pieter Lechner "Beneath Apple Dos", et je ne voudrais pas terminer sans redire, à mon tour, tout le bien qu'il faut penser de cet ouvrage. Pour les néophytes curieux de savoir ce qu'il y a réellement sur un disque, l'étude du chapitre 3 est passionnante.

N.D.L.R. : il est préférable de faire un FP après la sortie du programme, voire de 'rebooter' car le DOS est modifié.

Source COMPDISK.SCE
Assembleur LISA 2.5

```
;*************************************
;*                                   *
;*   PROGRAMME DE COMPARAISON GLOBALE *
;*      DE DISQUETTES 16 SECTEURS    *
;*                                   *
;*       C.I.VIVIER    2/1/86         *
;*                                   *
;*                          LISA 2.5 *
;*************************************
;
        NOG
;
;  ** ZONES DE TRAVAIL **
;
PTR     EPZ $0              ;POINTEURS POUR
       DUMP
PTR2    EPZ $2              ;ET COMPARAISON
TRACK   EPZ $4
DRIVE   EPZ $5
SCT     EPZ $6
WN     ;FENETRES
WNDB    EPZ $2
HPOS    EPZ $24             ;CURSEUR
VPOS    EPZ $25
CPT     EPZ $80
CPTB    EPZ $82
SVAD    EPZ $84             ;SAUV. NO SECT
RTN     EPZ $86             ;ADR RETOUR POUR
       PRINT
SLOT    EPZ $88
FLG     EPZ $89             ;AIGUILLEUR
SVPTR   EPZ $8A             ;SAUV. POINTEURS
SYST    EPZ $8E             ;IND. SYSTEME
PROLOG  EPZ $9D             ;PROLOG. A PATCH
       ER
RET     EPZ $8D             ;ASCII <RETURN>
ESC     EPZ $9B             ;ASCII <ESCAPE>
;
;  ** VECTEURS DU DOS **
;
VECPML  EQU $3E3
RWTS    EQU $3D9
DOS     EQU $3D0
;
;  ** MEMOIRES-TAMPONS **
;
INPUT   EQU $200
BUF1    EQU $200D           ;POUR DRIVE 1
LBUF1   EQU $4000           ;FIN BUF1
BUF2    EQU $4000
LBUF2   EQU $6000
BF1S    EQU $7000           ;BUFFERS DE
BF1P    EQU $7100           ;DENIBBILISATION
BF2S    EQU $8000
BF2P    EQU $8100
TRANSL  EQU $BA00           ;TABL.CONVERSION
       DOS
;
;  ** MEMOIRE-HARDWARE **
;
KBD     EQU $C000
KBDSTRB EQU $C010
DRV0EN  EQU $C08A           ;SELECT DR 1
DRV1EN  EQU $C08B           ;       DR 2
PHASEOFF EQU $C080          ;MOUVEMENTS DU
PHASEON EQU $C081           ;BRAS DE LECTEUR
MOTORON EQU $C089
MOTOROFF EQU $C088
Q6L     EQU $C08C           ;LECT DE NIBBLE
Q7L     EQU $C08E           ;PREP.LECTURE
;
;  ** ROUTINES MONITEUR ET APPLESOFT **
;
LINPRT  EQU $ED24           ;DECIMALISATION
PRBL2   EQU $F94A
TEXT    EQU $FB39
VTAB    EQU $FC22
HOME    EQU $FC58
WAIT    EQU $FCA8
RDCRAR  EQU $FD35
CROUT   EQU $FD8E
COUT    EQU $FDED
PRBYTE  EQU $FDDA
BELL    EQU $FF3A
;
        ORG $7FD
;
        JMP DEBUT
PML     DFS 1               ;INDEXATION DE
       LA PML
PSLOT   DFS 1
PDRV    DFS 1
PVOL    DFS 1
PTRK    DFS 1
PSEC    DFS 1
POCT    DFS 2
PBUF    DFS 2
PBID    DFS 2
PCMD    DFS 1
PERR    DFS 1
PVNF    DFS 1
PSLF    DFS 1
PDRF    DFS 1
;
;  **  DEBUT DU PROGRAMME  **
;
DEBUT   JSR HOME
        JSR PRINT
        INV '-----------------'
        HEX 8D
        INV '  COMPARATEUR DE  '
        HEX 8D
        INV '  DISQUES 16-SCT  '
        HEX 8D
        INV '-----------------'
        HEX 00
        JSR DEPATCH         ;RESET DES PROL.
       STANDARD
        LDA #5
        STA WNDT            ;FENETRE
        JSR HOME            ;AFFICHAGE 1ERES
       OPTIONS
        JSR PRINT
        ASC "- INSTALLEA LES DISQUES A COMPA
       RER"
        HEX 8D8D
        ASC "- PRECISER LE SYSTEME D'EXPLOIT
       ATION"
        HEX 8D8D
        ASC "     "
        INV "1"
        ASC " -> DOS"
        HEX 8D8D
        ASC "     "
        INV "2"
        ASC " -> PASCAL ET PRODOS"
        HEX 8D8D
        ASC "     "
        INV "3"
        ASC " -> CP/M"
        HEX 8D8D
        ASC "     "
        INV "4"
        ASC " -> INCONNU  (SECTEUR PHYSIQUE)
       "
        HEX 8D8D00
^1      BIT KBDSTRB
^2      LDA KBD             ;PREMIER CHOIX
        BPL <2
        CMP #$B0
        BCC <1
        CMP #$B5
        BCS <1
        EOR #$B0
        STA SYST            ;SAUV. SYSTEME
        JSR PRINT
        ASC "- PUIS, S'IL S'AGIT DE DISQUETT
       ES NON"
        HEX 8D8D
        ASC "      PROTEGEES FRAPPER <RETURN
       >"
        HEX 8D8D
        ASC "          SINON FRAPPER <ESC>"
        HEX DD
^3      BIT KBDSTRB
^4      LDA KBD
        BPL <4
        CMP #RET
        BNE >5
        JMP START
^5      CMP #ESC
        BNE <3
PROL    JSR HOME            ;AFFICHAGE POUR
       SAISIE PROLOGUES
        JSR PRINT
        ASC "LE PROGRAMME RECONNAIT LES DISQ
       UES AVEC"
        HEX 8D8D
        ASC "      LE PROLOGUE D'ADRESSE : "
        INV "D5AA96"
        HEX 8D8D
        ASC "      LE PROLOGUE DE DATA   : "
        INV "D5AA--"
        HEX 8D8D
        ASC "- S'IL EN EST AINSI, FRAPPER <R
       ETURN>"
        HEX 8D80
        ASC "- SINON ENTRER LES PROLOGUES"
        HEX 8D8D
        ASC "              D'ADRESSE: "
        INV "------"
        HEX 8D8D
        ASC "              DE DATA  : "
        INV "------"
        HEX 8D8D00
        LDA #19
        JSR TRAIT
        BIT KBDSTRB
^1      LDA KBD             ;CHOIX 2
        BPL <1
        CMP #RET
        BEQ START
        JSR SAIPRO          ;SAISI PROLOGUES
        LDA #21
        STA VPOS
        LDA #0
        STA HPOS
        JSR VTAB
        JSR PRINT
        ASC "CONFIRMEZ-VOUS ?  OUI -> <RETUR
       N>"
        HEX 8D8D
        ASC "                  NON -> <ESC>"
        HEX 00
^2      BIT KBDSTRB
^3      LDA KBD
        BPL <3
        CMP #RET
        BEQ >4
        CMP #ESC
        BNE <2
        JMP PROL
^4      JSR NUMPRO          ;CONVERSION HEXA
       PROLOG.
        JSR PATCH           ;PATCHES POUR PR
       OLOGUES
START   BIT KBDSTRB
        JSR PATSYS          ;PATCHES POUR SY
       STEME
        JSR HOME
        LDA #22
        JSR TRAIT
        JSR PRINT
        ASC "<ESPACE> POUR SUSPENSION ET REP
       RISE"
```

```
        HEX 00
        LDA #10
        STA VPOS
        JSR VTAB
        LDA #22
        STA WNDB
        LDA #0
        STA TRACK
        STA FLG
        JSR VECPML
        STY PTR
        STA PTR+1
        LDY #PSLOT          ;CHERCH PORT DSQ
        LDA (PTR),Y
        STA SLOT
        TAX
        LDA ORVIEN,X        ;CALAGE DU BRAS
      DE DR 2
        LDA MOTORON,X       ;SUR LA PISTE 0
        LDY #$50
TRACK0  LDA PHASEOFF,X
        TYA
        AND #$03
        ASL
        ORA SLOT
        TAX
        LDA PHASEON,X
        LDA #$56
        JSR WAIT
        DEY
        BPL TRACK0
        LDA SLOT
        TAX
        LDA MOTOROFF,X
;
;  ** EXPLORATION DES 35 PISTES **
;
BOUCLE  LDA #1
        STA DRIVE
        STA HPOS
        LDA FLG
        CMP #2
        BEQ >1
        LDA #10
        STA HPOS
        STA VPOS
        JSR VTAB
        JSR PRINT
        ASC "PISTES: "
        HEX 00
^1      LDA "$
        JSR COUT
        LDA TRACK
        JSR PRBYTE
        JSR DUMP            ;LECTURE D'UNE P
      ISTE I
        LDA #2
        STA DRIVE
        JSR DUMP            ;LECT. MEME PIST
      E II
        JSR COMPARE         ; --> COMPAR
^2      INC TRACK           ;PISTE SUIVANTE
        LDA TRACK
        CMP #$23            ;JUSQU'A 35EME
        BCC BOUCLE
        JSR BELL
        LDA FLG
        BNE >1
        JSR HOME
        LDA #10
        STA VPOS
        LDA #5
        STA HPOS
        JSR VTAB
        JSR PRINT
        BLK 'VOS DISQUETTES SONT IDENTIQUES'
        HEX 00
^1      BIT KBDSTRB
        JSR TEXT
        LDA #0
        STA HPOS
        LDA #23
        STA VPOS
        JSR VTAB
        JSR PRINT
        ASC "D'AUTRES DISQUETTES ?  --> <RET
      URN> "
        HEX 00
^2      LDA KBD
        BPL <2
        CMP #RET
        BNE >3
        JMP DEBUT
^3      JSR HOME            ;FIN PROGRAMME
        JMP DOS             ;RETOUR AU DOS
;
;  ** TRANSFERT EN MEMOIRE DES 2 PISTES **
;
DUMP    JSR VECPML          ;INSTALLATION
        STY PTR             ;PARAMETER LIST
      POUR RWTS
        STA PTR+1
        LDY #PTRK
        LDA TRACK
        STA (PTR),Y
        LDY #PCMD
        LDA #0              ;CDE "SEARCH"
        STA (PTR),Y
        LDY #PDRV
        LDA DRIVE
        STA (PTR),Y
        LDY #PVOL
        LDA #0
        STA (PTR),Y
        LDY #PVNF
        STA (PTR),Y
        JSR VECPML          ;CALL DE "RWTS"
        JSR RWTS
        LDA #0              ;APRES
        STA $48             ;UN CALL DE RWTS
        LDY #PSLOT
        LDA (PTR),Y
        TAX
        LDA MOTORON,X       ;TOURNE ENCORE P
      OUR LECT
        LDA Q7L,X           ;PREP. LECTURE
        LDA #0
        STA PTR
        LDA DRIVE           ;SET BUFFER EN
        ASL                 ;F(N) DU DRIVE
        ASL
        ASL
        ASL
        ASL
        STA PTR+1
        LDA #$E0            ;$20 INCR. AMENE
      RON7
        STA CPT             ;CPT A ZERO
        LDY #0
RD      LDA Q6L,X           ;LECTURE NIBBLE
        BPL RD
        CMP #$FF            ;JUSQU'A TROUVER
        BNE RD              ;2 FF AUTO-SYNC
RD1     LDA Q6L,X
        BPL RD1
        CMP #$FF
        BNE RD
RD2     LDA Q6L,X           ;LECT. JUSQU'A
        BPL RD2             ;FIN DES "FF"
        CMP #$FE
        BEQ RD2
        BNE RD4             ;DEBUT DE
RD3     LDA Q6L,X           ;DATA OU ADRESS
        BPL RD3
RD4     STA (PTR),Y         ;TRANSFERT
        INC PTR             ;TIMING CRITIQUE
```

Récapitulation 'COMPDISK'

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE COMPDISK,A$7FD,L$9BA

```
07FD- 4C 11 08
0800- A9 4C 8D 7A A5 8D 9E A5
0808- A9 84 8D 7B A5 8D 9F A5
0810- A9 20 58 FC 20 01 10 2D
0818- 2D 2D 2D 2D 2D 2D 2D 2D
0820- 2D 2D 2D 2D 2D 2D 2D 2D
0828- 2D 8D 20 20 03 0F 0D 10
0830- 01 12 01 14 05 15 12 20
0838- 04 05 20 20 8D 20 20 04
0840- 09 13 11 15 05 13 20 31
0848- 36 2D 13 03 14 20 20 8D
0850- 2D 2D 2D 2D 2D 2D 2D 2D
0858- 2D 2D 2D 2D 2D 2D 2D 2D
0860- 2D 2D 00 20 12 11 A9 05
0868- 85 22 20 58 FC 20 01 10
0870- AD A0 C9 CE D3 D4 C1 CC
0878- CC C5 D2 A0 CC C5 D3 A0
0880- C4 C9 D3 D1 D5 C5 D3 A0
0888- C1 A0 C3 CF CD D0 C1 D2
0890- C5 D2 8D 8D AD A0 D0 D2
0898- C5 C3 C9 D3 C5 D2 A0 CC
08A0- C5 A0 D3 D9 D3 D4 C5 CD
08A8- C5 A0 C4 A7 C5 D8 D0 CC
08B0- CF C9 D4 C1 D4 C9 CF CE
08B8- 8D 8D A0 A0 A0 A0 A0 31
08C0- A0 AD BE A0 C4 CF D3 8D
08C8- 8D A0 A0 A0 A0 A0 32 A0
08D0- AD BE A0 D0 C1 D3 C3 C1
08D8- CC A0 C5 D4 A0 D0 D2 CF
08E0- C4 CF D3 8D 8D A0 A0 A0
08E8- A0 A0 33 A0 AD BE A0 C3
08F0- D0 AF CD 8D 8D A0 A0 A0
08F8- A0 A0 34 A0 AD BE A0 C9
0900- CE C3 CF CE CE D5 A0 A0
0908- A8 D3 C5 C3 D4 C5 D5 D2
0910- A0 D0 C8 D9 D3 C9 D1 D5
0918- C5 A9 8D 8D 00 2C 10 C0
0920- AD 00 C0 10 FB C9 B0 90
0928- F4 C9 B5 B0 F0 49 B0 85
0930- 8E 20 01 10 AD A0 D0 D5
0938- C9 D3 AC A0 D3 A7 C9 CC
0940- A0 D3 A7 C1 C7 C9 D4 A0
0948- C4 C5 A0 C4 C9 D3 D1 D5
0950- C5 D4 D4 C5 D3 A0 CE CF
0958- CE 8D 8D A0 A0 A0 A0 A0
0960- A0 D0 D2 CF D4 C5 C7 C5
0968- C5 D3 A0 C6 D2 C1 D0 D0
0970- C5 D2 A0 BC D2 C5 D4 D5
0978- D2 CE BE 8D 8D A0 A0 A0
0980- A0 A0 A0 A0 A0 A0 A0 D3
0988- C9 CE CF CE A0 C6 D2 C1
0990- D0 D0 C5 D2 A0 BC C5 D3
0998- C3 BE 00 2C 10 C0 AD 00
09A0- C0 10 FB C9 8D D0 03 4C
09A8- 26 0B C9 9B D0 ED 20 58
09B0- FC 20 01 10 CC C5 A0 D0
09B8- D2 CF C7 D2 C1 CD CD C5
09C0- A0 D2 C5 C3 CF CE CE C1
09C8- C9 D4 A0 CC C5 D3 A0 C4
09D0- C9 D3 D1 D5 C5 D3 A0 C1
09D8- D6 C5 C3 8D 8D A0 A0 A0
09E0- A0 A0 A0 CC C5 A0 D0 D2
09E8- CF CC CF C7 D5 C5 A0 C4
09F0- A7 C1 C4 D2 C5 D3 D3 C5
09F8- A0 BA A0 04 35 01 01 39
0A00- 36 8D 8D A0 A0 A0 A0 A0
0A08- A0 CC C5 A0 D0 D2 CF CC
0A10- CF C7 D5 C5 A0 C4 C5 A0
0A18- C4 C1 D4 C1 A0 A0 A0 BA
0A20- A0 04 35 01 01 2D 2D 8D
0A28- 8D A0 A0 D3 A7 C9 CC A0
0A30- C5 CE A0 C5 D3 D4 A0 C1
0A38- C9 CE D3 C9 AC A0 C6 D2
0A40- C1 D0 D0 C5 D2 A0 BC D2
```

```
        BNE RD3             ;MAXI 31 CYCLES
        INC PTR+1
        INC CPT
        BNE RD3
        LDA MOTOROFF,X      ;$2000 NIBBLES
        RTS                 ;SONT TRANSFERES
;
;  ** COMPARAISON DES 16 SECTEURS **
;
COMPARE LDA #0              ;INITIALISE
        STA PTR             ;POINTEURS
        STA PTR2
        LDA /BUF1
        STA PTR+1
        LDA /BUF2
        STA PTR2+1
        LDY #0              ;ET CPTR DE SECT
        STY CPT
BALAY   LDA PTR+1
        CMP /LBUF1
        BCC >1
        JMP ERR             ;PROLOG.NON RECO
      NNUS
^1      JSR INCR
BAL.1   CMP #$D5            ;CHERCHE PROLOGU
        BNE BALAY
        JSR INCR
BAL.2   CMP #$AA
        BNE BALAY
        JSR INCR
BAL.3   CMP #$96            ;PROL. ADRESSE ?
        BEQ >2              ;OUI
        INC PTR+1           ;NON, + LOIN
        JMP BALAY
^2      LDX #5
^3      JSR INCR
        DEX
        BNE <3
        STA SVAD            ;SAUV NIBBLES
        JSR INCR            ;DONNANT NO SECT
        STA SVAD+1
^4      JSR INCR
BAL.4   CMP #$D5            ;CHERCHE PROL. D
      ATA
        BNE <4
        JSR INCR
BAL.5   CMP #$AA
        BNE <4
        JSR INCR            ;ON Y EST
        JSR SEARCH          ;CHERCHE HOMOLOG
      UE SUR II
        LDA #0              ;INIT POINT DR 2
        STA PTR2
        LDA /BUF2
        STA PTR2+1
        LDY #0
        INC CPT             ;SECT SUIV
        LDA CPT
        CMP #$10            ;JUSQU'AU 16EME
        BCC BALAY
        RTS
;
;  ** RECHERCHE DU SECTEUR A COMPARER **
;
SEARCH  LDA KBD             ;SUSPENDRE LE
        BPL >5              ;TRAITEMENT
        BIT KBDSTRB
^6      LDA KBD             ;REPRENDRE
        BPL <6
        BIT KBDSTRB
^5      LDA PTR2+1
        CMP /LBUF2
        BCC >7
        JSR ERR             ;PISTE ILLISIBLE
        PLA                 ;RETOUR DANS BOU
      CLE PRINCIPALE
        PLA
        RTS
^7      JSR INCR2
SRC.1   CMP #$D5
        BNE <5
        JSR INCR2
SRC.2   CMP #$AA
        BNE <5
        JSR INCR2
SRC.3   CMP #$96
        BEQ >2
        INC PTR2+1
        JMP <5
^2      LDX #5
^3      JSR INCR2
        DEX
        BNE <3              ;ON EST SUR UN S
      ECTEUR
        CMP SVAD            ;EST-CE LE BON ?
        BEQ >5
^4      INC PTR2+1
        JMP <5              ;NON -> SUIVANT
^5      JSR INCR2
        CMP SVAD+1
        BEQ >6
        JMP <4
^6      JSR INCR2
SRC.4   CMP #$D5
        BNE <6
        JSR INCR2
SRC.5   CMP #$AA
        BNE <6
        JSR INCR2           ;LA COMP.PEUT CO
      MMENCER
        LDA #0
        STA CPTB+1
        LDA #$AA            ;POUR $156 NIBBL
        STA CPTB
        LDX #3              ;SAUV.ADR.DEBUT
^1      LDA PTR,X           ;DATA DES 2 SECT
        STA SVPTR,X
        DEX
        BPL <1
;
;  ** COMPARAISON **
;
TEST    LDA (PTR),Y
        CMP (PTR2),Y
        BNE DIF             ;DESACCORD
        JSR INCS
        JSR INCS2
        INC CPTB
        BNE TEST
        INC CPTB+1
        LDA CPTB+1
        CMP #2
        BCC TEST
        RTS                 ;$156 NIBBLES =
INCR    LDA (PTR),Y
INCS    INC PTR             ;ROUTINES D'INCR
        BNE >1              ;DE POINTEURS
        INC PTR+1
^1      RTS
INCR2   LDA (PTR2),Y
INCS2   INC PTR2
        BNE <1
        INC PTR2+1
        RTS
DIF     LDA SVPTR           ;TROUVE POSIT.
        STA PTL+1           ;DENIBBILISE LE
        LDA SVPTR+1         ;SECTEUR I
        STA PTH+1           ;SELON ROUTINE D
      U DOS
        LDA /BF1S
        STA ST1+2
        LDA /BF1P
        STA ST2+2
        JSR TRF             ;TRANSFERT EN BU
      FFERS
        LDA SVPTR+2         ;PRIMAIRE ET SEC
```

```
0A48- C5 D4 D5 D2 CE BE 8D 8D
0A50- AD A0 D3 C9 CE CF CE A0
0A58- C5 CE D4 D2 C5 D2 A0 CC
0A60- C5 D3 A0 D0 D2 CF CC CF
0A68- C7 D5 C5 D3 8D 8D A0 A0
0A70- A0 A0 A0 A0 A0 A0 A0 A0
0A78- C4 A7 C1 C4 D2 C5 D3 D3
0A80- C5 BA A0 2D 2D 2D 2D 2D
0A88- 2D 8D 8D A0 A0 A0 A0 A0
0A90- A0 A0 A0 A0 A0 C4 C5 A0
0A98- C4 C1 D4 C1 A0 A0 BA A0
0AA0- 2D 2D 2D 2D 2D 2D 8D 8D
0AA8- 00 A9 13 20 29 10 2C 10
0AB0- C0 AD 00 C0 10 FB C9 8D
0AB8- F0 6C 20 3D 10 A9 15 85
0AC0- 25 A9 00 85 24 20 22 FC
0AC8- 20 01 10 C3 CF CE C6 C9
0AD0- D2 CD C5 DA AD D6 CF D5
0AD8- D3 A0 BF A0 A0 CF D5 C9
0AE0- A0 AD BE A0 BC D2 C5 D4
0AE8- D5 D2 CE BE 8D 8D A0 A0
0AF0- A0 A0 A0 A0 A0 A0 A0 A0
0AF8- A0 A0 A0 A0 A0 A0 A0 A0
0B00- CE CF CE A0 AD BE A0 BC
0B08- C5 D3 C3 BE 00 2C 10 C0
0B10- AD 00 C0 10 FB C9 8D F0
0B18- 07 C9 9B D0 F0 4C AE 09
0B20- 20 83 10 20 AD 10 2C 10
0B28- C0 20 D6 10 20 58 FC A9
0B30- 16 20 29 10 20 01 10 BC
0B38- C5 D3 D0 C1 C3 C5 BE A0
0B40- D0 CF D5 D2 A0 D3 D5 D3
0B48- D0 C5 CE D3 C9 CF CE A0
0B50- C5 D4 A0 D2 C5 D0 D2 C9
0B58- D3 C5 00 A9 0A 85 25 20
0B60- 22 FC A9 16 85 23 A9 00
0B68- 85 04 85 89 20 E3 03 84
0B70- 00 85 01 A0 01 B1 00 85
0B78- 88 AA BD 8B C0 BD 89 C0
0B80- A0 50 BD 80 C0 98 29 03
0B88- 0A 05 88 AA BD 81 C0 A9
0B90- 56 20 A8 FC 88 10 EB A5
0B98- 88 AA BD 88 C0 A9 01 85
0BA0- 05 85 24 A5 89 C9 02 F0
0BA8- 15 A9 0A 85 24 85 25 20
0BB0- 22 FC 20 01 10 D0 C9 D3
0BB8- D4 C5 D3 BA A0 00 A9 A4
0BC0- 20 ED FD A5 04 20 DA FD
0BC8- 20 60 0C A9 02 85 05 20
0BD0- 60 0C 20 DD 0C E6 04 A5
0BD8- 04 C9 23 90 C0 20 3A FF
0BE0- A5 89 D0 30 20 58 FC A9
0BE8- 0A 85 25 A9 05 85 24 20
0BF0- 22 FC 20 01 10 56 4F 53
0BF8- 60 44 49 53 51 55 45 54
0C00- 54 45 53 60 53 4F 4E 54
0C08- 60 49 44 45 4E 54 49 51
0C10- 55 45 53 00 2C 10 C0 20
0C18- 39 FB A9 00 85 24 A9 17
0C20- 85 25 20 22 FC 20 01 10
0C28- C4 A7 C1 D5 D4 D2 C5 D3
0C30- A0 C4 C9 D3 D1 D5 C5 D4
0C38- D4 C5 D3 A0 BF A0 A0 AD
0C40- AD BE A0 BC D2 C5 D4 D5
0C48- D2 CE BE A0 A0 00 AD 00
0C50- C0 10 FB C9 8D D0 03 4C
0C58- 11 08 20 58 FC 4C D0 03
0C60- 20 E3 03 84 00 85 01 A0
0C68- 04 A5 04 91 00 A0 0C A9
0C70- 00 91 00 A0 02 A5 05 91
0C78- 00 A0 03 A9 00 91 00 A0
0C80- 0E 91 00 20 E3 03 20 D9
0C88- 03 A9 00 85 48 A0 01 B1
0C90- 00 AA BD 89 C0 BD 8E C0
0C98- A9 00 85 00 A5 05 0A 0A
0CA0- 0A 0A 0A 85 01 A9 E0 85
0CA8- 80 A0 00 BD 8C C0 10 FB
0CB0- C9 FF D0 F7 BD 8C C0 10
0CB8- FB C9 FF D0 EE BD 8C C0
0CC0- 10 FB C9 FF F0 F7 D0 05
0CC8- BD 8C C0 10 FB 91 00 E6
```

```
        ONDAIRE
          STA PTL+1
          LDA SVPTR+3
          STA PTH+1
          LDA /BF2S
          STA ST1+2
          LDA /BF2P
          STA ST2+2
          JSR TRP              ;IDEM POUR SECTE
        UR II
          LDY #0
^1        LDX #0               ;TRANSFORME
^2        LDA BF1P,Y           ;NIBBLES EN OCT
          LSR BF1S,X
          ROL
          LSR BF1S,X
          ROL
          STA BF1P,Y
          INY
          BEQ >3
          INX
          CPX #$56
          BEQ <1
          BNE <2
^3        LDX #0               ;ON COMMENCE DE
        MEME POUR
^4        LDA BF2P,Y           ;LE SECTEUR II
          LSR BF2S,X
          ROL
          LSR BF2S,X
          ROL
          CMP BF1P,Y           ;JUSQU'A CE QU'O
        N TROUVE
          BNE >6               ;LA DIFFERENCE
          INY
          BEQ >5
          INX
          CPX #$56
          BEQ <3
          BNE <4
^5        JMP NEXT
^6        TYA
          PHA                  ; SAUV. DE LA PO
        SITION
          LDA FLG              ;TTMENT DE LA DI
        FFERENCE
          CMP #2
          BEQ PRERR            ;1 DIFFERENCE A
        DEJA
          LDA #2               ;ETE VUE
          STA FLG
          JSR HOME             ;AFFICHAGE ENTET
          JSR PRINT
          ASC "IL Y A DIFFERENCE POUR   "
          ASC "(SYST:"
LISYS     ASC "        "       ;
          HEX 8D8D             ;
          INV "PISTE  SECTEUR  POSITION"
          HEX 00
          LDA SYST             ;AFFICHAGE DE "B
        LOC"
          AND #$02             ;SI SYST = 2 OU
        3
          BEQ >1
          JSR PRINT
          ASC "   BLOC"
          HEX 00
^1        LDA #9
          STA WNDT
          JSR HOME
PRERR     LDX #1               ;AFFICHAGE DE
          STX HPOS
          JSR HEXA
          LDA TRACK            ;LA PISTE
          JSR PRBYTE
          LDX #5
          JSR PRBL2
          JSR HEXA
          SEC
          LDA SVAD
          ROL
          AND SVAD+1           ;PUIS DU SECT.LO
        GIQUE
          TAX
ADTSYS    LDA $1000,X          ;ZONE A PATCHER
          STA SCT              ;EN F(N) DU SYS
          JSR PRBYTE
          LDX #6
          JSR PRBL2
          JSR HEXA
          PLA                  ;RECUPERATION DE
        POSITION
          JSR PRBYTE
          LDX #6
          JSR PRBL2
          LDA SYST             ;CALCUL  NO BLOC
          CMP #2
          BEQ BLKPAS
          CMP #3
          BEQ BLKCPM
          JSR CROUT
NEXT      CLC
          LDA SVPTR            ;ON AMENE LE PON
        TEUR I
          ADC #$56             ;SUR LE SECT. I
        SUIVANT
          STA PTR
          LDA SVPTR+1
          ADC #$00
          STA PTR+1
          RTS
BLKPAS    LSR SCT              ;POUR PASCAL ET
        PRODOS
          PHP
          LDA TRACK
          ASL
          ASL
          ASL
          TAX
          LDA #0
          ADC #0
          PHA
          CLC
          TXA
          ADC SCT
          TAX
          PLA
          JSR LINPRT           ;AFFICH  DECIMAL
          LDA ",
          JSR COUT
          PLP
          LDA #0
          ADC #$B1
          JSR COUT             ;PARTIE DU BLOC
        1 OU 2
          JSR CROUT
          JMP NEXT
BLKCPM    ASL TRACK            ;IDEM SI CP/M
          ASL TRACK
          LDA SCT
          LSR
          LSR
          CLC
          ADC TRACK
          TAX
          LSR TRACK
          LSR TRACK
          LDA #0
          JSR LINPRT
          LDA ",
          JSR COUT
          CLC
          LDA SCT
          AND #$03
          ADC #$B1
          JSR COUT
          JSR CROUT
          JMP NEXT
;
```

```
0CD0- 00 D0 F5 E6 01 E6 80 D0
0CD8- EF BD 88 C0 60 A9 00 85
0CE0- 00 85 02 A9 20 85 01 A9
0CE8- 40 85 03 A0 00 84 80 A5
0CF0- 01 C9 40 90 03 4C 94 0F
0CF8- 20 D4 0D C9 D5 D0 F0 20
0D00- D4 0D C9 AA D0 E9 20 D4
0D08- 0D C9 96 F0 05 E6 01 4C
0D10- EF 0C A2 05 20 D4 0D CA
0D18- D0 FA 85 84 20 D4 0D 85
0D20- 85 20 D4 0D C9 D5 D0 F9
0D28- 20 D4 0D C9 AA D0 F2 20
0D30- D4 0D 20 48 0D A9 00 85
0D38- 02 A9 40 85 03 A0 00 E6
0D40- 80 A5 80 C9 10 90 A8 60
0D48- AD 00 C0 10 0B 2C 10 C0
0D50- AD 00 C0 10 FB 2C 10 C0
0D58- A5 03 C9 60 90 06 20 94
0D60- 0F 68 68 60 20 DD 0D C9
0D68- D5 D0 ED 20 DD 0D C9 AA
0D70- D0 E6 20 DD 0D C9 96 F0
0D78- 05 E6 03 4C 58 0D A2 05
0D80- 20 DD 0D CA D0 FA C5 84
0D88- F0 05 E6 03 4C 58 0D 20
0D90- DD 0D C5 85 F0 03 4C 8A
0D98- 0D 20 DD 0D C9 D5 D0 F9
0DA0- 20 DD 0D C9 AA D0 F2 20
0DA8- DD 0D A9 00 85 83 A9 AA
0DB0- 85 82 A2 03 B5 00 95 8A
0DB8- CA 10 F9 B1 00 D1 02 D0
0DC0- 25 20 D6 0D 20 DF 0D E6
0DC8- 82 D0 F0 E6 83 A5 83 C9
0DD0- 02 90 E8 60 B1 00 E6 00
0DD8- D0 02 E6 01 60 B1 02 E6
0DE0- 02 D0 F9 E6 03 60 A5 8A
0DE8- 8D 60 0F A5 8B 8D 6A 0F
0DF0- A9 70 8D 7F 0F A9 71 8D
0DF8- 8F 0F 20 5E 0F A5 8C 8D
0E00- 60 0F A5 8D 8D 6A 0F A9
0E08- 80 8D 7F 0F A9 81 8D 8F
0E10- 0F 20 5E 0F A0 00 A2 00
0E18- B9 00 71 5E 00 70 2A 5E
0E20- 00 70 2A 99 00 71 C8 F0
0E28- 07 E8 E0 56 F0 E8 D0 E8
0E30- A2 00 B9 00 81 5E 00 80
0E38- 2A 5E 00 80 2A D9 00 71
0E40- D0 0D C8 F0 07 E8 E0 56
0E48- F0 E6 D0 E6 4C FC 0E 98
0E50- 48 A5 89 C9 02 F0 64 A9
0E58- 02 85 89 20 58 FC 20 01
0E60- 10 C9 CC A0 D9 A0 C1 A0
0E68- C4 C9 C6 C6 C5 D2 C5 CE
0E70- C3 C5 A0 D0 CF D5 D2 A0
0E78- A0 A0 A8 D3 D9 D3 D4 BA
0E80- A0 A0 A0 A0 A0 A0 A0 A0
0E88- 8D 8D 10 09 13 14 05 20
0E90- 20 13 05 03 14 05 15 12
0E98- 20 20 10 0F 13 09 14 09
0EA0- 0F 0E 00 A5 8E 29 02 F0
0EA8- 0B 20 01 10 A0 A0 A0 C2
0EB0- CC CF C3 00 A9 09 85 22
0EB8- 20 58 FC A2 01 86 24 20
0EC0- 24 10 A5 04 20 DA FD A2
0EC8- 05 20 4A F9 20 24 10 38
0ED0- A5 84 2A 25 85 AA BD 00
0ED8- 10 85 06 20 DA FD A2 06
0EE0- 20 4A F9 20 24 10 68 20
0EE8- DA FD A2 06 20 4A F9 A5
0EF0- 8E C9 02 F0 15 C9 03 F0
0EF8- 3B 20 8E FD 18 A5 8A 69
0F00- 56 85 00 A5 8B 69 00 85
0F08- 01 60 46 06 08 A5 04 0A
0F10- 0A 0A AA A9 00 69 00 48
0F18- 18 8A 65 06 AA 68 20 24
0F20- ED A9 AC 20 ED FD 28 A9
0F28- 00 69 B1 20 ED FD 20 8E
0F30- FD 4C FC 0E 06 04 06 04
0F38- A5 06 4A 4A 18 65 04 AA
0F40- 46 04 46 04 A9 00 20 24
```

```
;  ** ROUTINES DIVERSES **
;
TRF     CLC                     ;PREP DE LA DENI
        BBILIS.
PTL     LDA #00                 ;PATCH DE LA ROU
        TINE
        STA LD1+1               ;POUR QU'ELLE SO
        IT
        ADC #$56                ;COMMUNE AUX 2 D
        R
        STA LD2+1
PTH     LDA #00
        STA LD1+2
        ADC #0
        STA LD2+2
        LDX #0
        LDA #0
LD1     LDY $1000,X
        EOR TRANSL,Y
ST1     STA $1000,X
        INX
        CPX #$56
        BNE LD1
        LDX #0
LD2     LDY $1000,X             ;CONVERSION
        EOR TRANSL,Y            ;EOR DES NIBBLES
ST2     STA $1000,X
        INX
        BNE LD2
        RTS
ERR     LDA WNDT                ;SORTIE MESSAGE
        PHA                     ;PISTE ILLISIBLE
        LDA VPOS
        PHA
        LDA #0
        STA WNDT
        LDA #1
        STA VPOS
        LDA #20
        STA HPOS
        JSR VTAB
        JSR PRINT
        ASC "PISTE $"
        HEX 00
        LDA TRACK
        JSR PRBYTE
        JSR PRINT
        ASC " DRIVE "
        HEX 00
        LDA PTR+1
        CMP /BUF2
        BCC >1
        LDA "1
        BNE >2
^1      LDA "2
^2      JSR COUT
        JSR CROUT
        LDA #24
        STA WPOS
        JSR PRINT
        ASC "ILLISIBLE"
        HEX 00
        JSR BELL
        JSR BELL
        LDA #1
        STA FLG
        PLA
        STA VPOS
        LDA #0
        STA HPOS
        JSR VTAB
        PLA
        STA WNDT
        LDY #0
        RTS
PRINT   PLA
        STA RTN
        PLA
        STA RTN+1
        LDY #1
^1      LDA (RTN),Y
        BEQ >2
        JSR COUT
        INY
        BNE <1
^2      CLC
        TYA
        ADC RTN
        STA RTN
        LDA RTN+1
        ADC #0
        PHA
        LDA RTN
        PHA
        LDY #0
        RTS
HEXA    LDA "$
        JMP COUT
TRAIT   STA VPOS                ;TIRER UN TRAIT
        LDA #0
        STA HPOS
        JSR VTAB
        LDA "-
        LDX #39
^3      JSR SEDED
        DEX
        BPL <3
        RTS
SAIPRO  PHA                     ;SAISIE PROLOGUE
        S
        LDA #21                 ;TRAITT DU
        STA HPOS                ;1ER CARAC SAISI
        LDA #15
        STA VPOS                ;SI <RETURN>
        JSR VTAB
        PLA
        STA INPUT
        AND #$3F                ;AFICH EN INVERS
        JSR COUT
        BIT KBDSTRB
        LDX #1
^4      JSR RDCHAR              ;SAISIE 5 AUTRES
        STA INPUT,X             ;DIGITS PROL ADR
        AND #$3F
        JSR COUT
        INX
        CPX #6
        BCC <4
        LDA #17
        STA VPOS
        LDA #21
        STA HPOS
        JSR VTAB
^5      JSR RDCHAR              ;PUIS PROL DATA
        STA INPUT,X
        AND #$3F
        JSR COUT
        INX
        CPX #12
        BCC <5
        RTS
NUMPRO  LDY #5                  ;CONVERSION HEXA
        LDX #11
^1      JSR NUM
        STA PROLOG,Y
        DEX
        JSR NUM
        ASL
        ASL
        ASL
        ASL
        CLC
        ADC PROLOG,Y
        STA PROLOG,Y            ;ET SAUV. EN PRO
        LOG
        DEX
        DEY
        BPL <1
        RTS
```

```
0F48- ED A9 AC 20 ED FD 18 A5
0F50- 06 29 03 69 B1 20 ED FD
0F58- 20 8E FD 4C FC 0E 18 A9
0F60- 00 8D 78 0F 69 56 8D 88
0F68- 0F A9 00 8D 79 0F 69 00
0F70- 8D 89 0F A2 00 A9 00 BC
0F78- 00 10 59 00 BA 9D 00 10
0F80- E8 E0 56 D0 F2 A2 00 BC
0F88- 00 10 59 00 BA 9D 00 10
0F90- E8 D0 F4 60 A5 22 48 A5
0F98- 25 48 A9 00 85 22 A9 01
0FA0- 85 25 A9 14 85 24 20 22
0FA8- FC 20 01 10 D0 C9 D3 D4
0FB0- C5 A0 A4 00 A5 04 20 DA
0FB8- FD 20 01 10 A0 C4 D2 C9
0FC0- D6 C5 A0 00 A5 01 C9 40
0FC8- 90 04 A9 B1 D0 02 A9 B2
0FD0- 20 ED FD 20 8E FD A9 18
0FD8- 85 24 20 01 10 C9 CC CC
0FE0- C9 D3 C9 C2 CC C5 00 20
0FE8- 3A FF 20 3A FF A9 01 85
0FF0- 89 68 85 25 A9 00 85 24
0FF8- 20 22 FC 68 85 22 A0 00
1000- 60 68 85 86 68 85 87 A0
1008- 01 B1 86 F0 06 20 ED FD
1010- C8 D0 F6 18 98 65 86 85
1018- 86 A5 87 69 00 48 A5 86
1020- 48 A0 00 60 A9 A4 4C ED
1028- FD 85 25 A9 00 85 24 20
1030- 22 FC A9 AD A2 27 20 ED
1038- FD CA 10 FA 60 48 A9 15
1040- 85 24 A9 0F 85 25 20 22
1048- FC 68 8D 00 02 29 3F 20
1050- ED FD 2C 10 C0 A2 01 20
1058- 35 FD 9D 00 02 29 3F 20
1060- ED FD E8 E0 06 90 F0 A9
1068- 11 85 25 A9 15 85 24 20
1070- 22 FC 20 35 FD 9D 00 02
1078- 29 3F 20 ED FD E8 E0 0C
1080- 90 F0 60 A0 05 A2 0B 20
1088- A1 10 99 90 00 CA 20 A1
1090- 10 0A 0A 0A 0A 18 79 90
1098- 00 99 90 00 CA 88 10 E7
10A0- 60 BD 00 02 49 B0 C9 0A
10A8- 90 02 E9 67 60 A5 90 8D
10B0- FC 0C 8D 68 0D A5 91 8D
10B8- 03 0D 8D 6F 0D A5 92 8D
10C0- 0A 0D 8D 76 0D A5 93 8D
10C8- 25 0D 8D 9D 0D A5 94 8D
10D0- 2C 0D 8D A4 0D 60 A2 00
10D8- BD A3 11 C5 8E F0 07 E8
10E0- E8 E8 E8 E8 D0 F2 BD A4
10E8- 11 8D D7 0E BD A5 11 8D
10F0- D8 0E BD A6 11 8D 05 11
10F8- 8D 09 11 BD A7 11 8D 06
1100- 11 8D 0A 11 AD 00 10 AA
1108- BD 00 10 9D 7F 0E CA D0
1110- F7 60 A9 D5 8D FC 0C 8D
1118- 25 0D 8D 68 0D 8D 9D 0D
1120- A9 AA 8D 03 0D 8D 2C 0D
1128- 8D 6F 0D 8D A4 0D A9 96
1130- 8D 0A 0D 8D 76 0D A2 07
1138- A9 A0 9D 80 0E CA 10 FA
1140- 60 00 07 0E 06 0D 05 0C
1148- 04 0B 03 0A 02 09 01 08
1150- 0F 00 08 01 09 02 0A 03
1158- 0B 04 0C 05 0D 06 0E 07
1160- 0F 00 0B 06 01 0C 07 02
1168- 0D 08 03 0E 09 04 0F 0A
1170- 05 00 01 02 03 04 05 06
1178- 07 08 09 0A 0B 0C 0D 0E
1180- 0F 07 A0 A0 C4 CF D3 A0
1188- A9 08 D0 C1 D3 AF D0 D2
1190- CF A9 07 A0 C3 D0 AF CD
1198- A0 A9 08 C9 CE C3 CF CE
11A0- CE D5 A9 01 41 11 81 11
11A8- 02 51 11 89 11 03 61 11
11B0- 92 11 04 71 11 9A 11
```

```
NUM     LDA INPUT,X
        EOR #$B0
        CMP #$0A
        BCC >2
        SBC #$67
^2      RTS
PATCH   LDA PROLOG          ;ROUTINE PATCH
        STA BAL.1+1         ;POUR PROLOGUES
        STA SRC.1+1
        LDA PROLOG+1
        STA BAL.2+1
        STA SRC.2+1
        LDA PROLOG+2
        STA BAL.3+1
        STA SRC.3+1
        LDA PROLOG+3
        STA BAL.4+1
        STA SRC.4+1
        LDA PROLOG+4
        STA BAL.5+1
        STA SRC.5+1
        RTS
PATSYS  LDX #0              ;PATCH TABLE DE
^1      LDA TSYS,X          ;CORRESP.SECT.PH
       YS
        CMP SYST            ;ET SECT.LOGIQUE
        BEQ PATS
        INX
        INX
        INX
        INX
        INX
        BNE <1
PATS    LDA TSYS+1,X
        STA ADTSYS+1
        LDA TSYS+2,X
        STA ADTSYS+2
        LDA TSYS+3,X
        STA TRLIB+1
        STA TRLIB2+1
        LDA TSYS+4,X
        STA TRLIB+2
        STA TRLIB2+2
TRLIB   LDA $1000           ;TRANSFERT DU LI
       BELLE
        TAX                 ;DE SYSTEME
TRLIB2  LDA $1000,X
        STA LISYS-1,X
        DEX
        BNE TRLIB2
        RTS
DEPATCH LDA #$05            ;REMISE A VAL. S
TNDO
        STA BAL.1+1         ;POUR PROLOGUES
        STA BAL.4+1
        STA SRC.1+1
        STA SRC.4+1
        LDA #$AA
        STA BAL.2+1
        STA BAL.5+1
        STA SRC.2+1
        STA SRC.5+1
        LDA #$96
        STA BAL.3+1
        STA SRC.3+1
        LDX #7              ;REMISE A SPACES
POUR
        LDA "               ;LIBELLE SYSTEME
^1      STA LISYS,X
        DEX
        BPL <1
        RTS
TDOS    HEX 00070E060D050C04
        HEX 0B030A020901080F
TPAS    HEX 00080109020A030B
        HEX 040C050D060E070F
TCPM    HEX 000B06010C07020D
        HEX 08030E09040F0A05
TPHY    HEX 0001020304050607
        HEX 08090A0B0C0D0E0F
;
LIBDOS  STR "  DOS )"
LIBPAS  STR "PAS/PRO)"
LIBCPM  STR " CP/M )"
LIBINC  STR "INCONNU)"
;
TSYS    HEX D1              ;TABLE D'ACCES A
       UX
        ADR TDOS            ;TABLES PRECEDEN
       TES
        ADR LIBDOS          ;D'APRES LE NO
        HEX 02              ;DE SYSTEME CHOI
       SI
        ADR TPAS
        ADR LIBPAS
        HEX 03
        ADR TCPM
        ADR LIBCPM
        HEX 04
        ADR TPHY
        ADR LIBINC
        END
```

Paddle, Souris, et Apple //c

Il est possible sur l'Apple //c d'utiliser la souris en lieu et place des paddles. La méthode est la suivante :

- En mode direct, taper : PR£4, puis CTRL-A, puis PR£0.
- En mode programme : PRINT CHR$(4) "PR£4" : PRINT CHR$(1) : PRINT CHR$(4) "PR£0"

A titre d'essai, vous pouvez utiliser le programme suivant :

```
10 PRINT CHR$(4) "PR£4" : PRINT CHR$(1) : PRINT CHR$(4) "PR£0"
20 PRINT PDL(0), PDL(1) : GOTO 20
```

La souris s'utilise alors comme un joystick. Ceci fonctionne avec les programmes Basic ; en revanche, la plupart des programmes commerciaux utilisant directement les adresses des paddles, ne sauront pas se servir de la souris...

Vous avez un Apple IIe avec Chat Mauve ou un //c ?
Vous disposez de Pascal 1.2 ?

Profitez de l'interpréteur **COGO**

Par Nicolas Montsarrat

Il s'agit d'un système graphique double-haute résolution écrit en Pascal.
COGO vous permet de manipuler des graphiques grâce à un langage de description des objets - points, angles- et à l'emploi de fonctions primitives de manipulation très puissantes : cercle, tangente, intersections, parallèles, etc. Il est ainsi possible de tracer des grilles, des cercles, des segments de droite, des tangentes communes à deux cercles, de calculer des distances, des angles...

L'éditeur permet une saisie rapide du langage. Une instruction COGO peut-être exécutée dès la saisie pour faciliter la mise au point, ou au sein d'un programme.

MAINTENANT EN DOUBLE HAUTE RÉSOLUTION SUR APPLE //C

Ce programme, destiné à résoudre des problèmes de géométrie plane, comporte des instructions de stockage sur fichier afin de permettre la reprise d'un calcul.

150,00 F TTC, franco
Bon de commande page 74

Le formateur de ProDOS désassemblé

Bruno Fénart

Par manque de place, le "kernel" (noyau) de ProDOS ne comporte pas de formateur. Contrairement au DOS 3.3, pour formater une disquette en ProDOS, il faut utiliser un programme spécifique. Il existe deux versions des utilitaires 'système' de ProDOS (FILER), plus un utilitaire spécial pour l'Apple //c.

La partie désassemblée ici concerne le formateur physique de la version la plus récente. Les deux versions de FILER diffèrent peu ; toutefois, l'emplacement du formateur a été déplacé de $7C00 à $7E00. De plus, un bug qui l'empêchait de fonctionner sur des lecteurs de disquettes trop lents ou trop rapides ("DISK II TOO FAST" ou "DISK II TOO SLOW") a été corrigé.

Les lecteurs pourront comparer ce formateur avec celui du DOS 3.3 (voir RWTS désassemblée Pom's n° 21). Ils pourront s'apercevoir que la principale différence entre les deux est que ProDOS détecte quand l'erreur provient de la vitesse de rotation du lecteur. Ils pourront également voir comment modifier les valeurs de GAPMIN et GAPMAX si leur lecteur de disquettes est trop lent ou trop rapide pour le formatage ProDOS.

Source 'FORMAT.SCE0'

Assembleur ProCode

```
0   *_____________________________________
1   *                                    *
2   *                                    *
3   *                                    *
4   *     FORMATEUR  du  FILER  de  ProDOS  *
5   *   Copyright Apple Computer JUN/DEC 84  *
6   *                                    *
7   *  Désassemblage  Bruno Fénart  10/SEP/85  *
8   *                                    *
9   *____________________________________*
10  *____________________________________*
11
12  * Formatage physique 16 secteurs DISK II
13  * Pour le FILER de SEP 83 le début est en $7800
14  * La copie du boot (blocs 0 et 1) se trouve en $65D3.$69D2
15  * ou en $6425.$6824 pour l'ancienne version
16
17  * Entrée: A = n° lecteur = n° slot x $10 + $80 si drive 2
18  *         VOLUME  ($DE)  = n° du volume formaté
19  * Sortie: Erreur        -> carry set   et A = n° erreur
20  *         Pas d'erreur -> carry clear et A = 0
21
22          ORG  $7A00       ;ORG $7800 pour l'ancienne version
23
24  *  Variables en page zéro
25  *  ----------------------
26
27  CST.AA   EQU  $D0        ;Constante $AA. Temporise 1 cycle
28  PIST.NB  EQU  $D1        ;n° piste à formater (init.= 0)
29  SECT.NB  EQU  $D2        ;n° secteur à formater (init.= 0)
30  VOL.NB   EQU  $D3        ;n° du volume (Init.= VOLUME)
31  GAP3     EQU  $D4        ;Nbre sync. bytes (init.=GAPMAX+2)
32  COMPTDAT EQU  $D5        ;Compteur des données lues
33  ESSAINB  EQU  $D6        ;Nombre d'essai par piste (Max=3)
34  TABLADR  EQU  $D7        ;Table de CHCKSM,SECT,PIST,VOL
35  SECTEUR  EQU  TABLADR+1  ;n° du secteur lu
36  COMPT0   EQU  $D9        ;Compte 256 DELAY. Non initialisé
37  COMPT1   EQU  $DA        ;Décompte 1 s. Initialisé à $D7
38  TAMPON1  EQU  $DB        ;Tampon pour le checksum (init.=0)
39  COMPTEUR EQU  $DC        ;Essais de lecture (init.= $FC)
40  TAMPON2  EQU  $DD        ;Tampon codage 4 et 4
41  VOLUME   EQU  $DE        ;Entrée: n° du volume
42
43  *  Commutateurs du lecteur de disquettes
44  *  -------------------------------------
45
46  DRVS0    EQU  $C080      ;Phase 0
47  DRVOFF   EQU  $C088      ;Arrêt
48  DRVON    EQU  $C089      ;Marche
49  DRVSL1   EQU  $C08A      ;Drive 1
50  DRVSL2   EQU  $C08B      ;Drive 2
51  DRVRD    EQU  $C08C      ;Lecture
52  DRVWR    EQU  $C08D      ;Ecriture
53  DRVRDM   EQU  $C08E      ;Mode lecture
54  DRVWRM   EQU  $C08F      ;Mode écriture
55
56
57  * MACROS de lecture et d'écriture
58  * --------------------------------
59
60          DO   0
61  READ    MAC
62          LDA  DRVRD,X     ;Lecture d'un octet
63          BPL  *-3         ;Lit tant que positif
64          EOM
65  WRITE   MAC              ;Ecriture sur 9 cycles
66                           ; (A inchangé, Carry modifié):
67          STA  DRVWR,X     ;Charge l'octet à écrire  (5)
68          CMP  DRVRD,X     ;L'écrit sur le disque    (9)
69          EOM
70  WRITEBIS MAC             ;Ecriture sur 9 cycles
71                           ; (A modifié, Carry inchangée):
72          STA  DRVWR,X     ;Charge l'octet à écrire  (5)
73          LDA  DRVRD,X     ;L'écrit sur le disque    (9)
74          EOM
75  WRSTATUS MAC             ;Test et initialise l'écriture
76                           ; (indispensable avant d'écrire):
77          LDA  DRVWR,X     ;Test si le disque est protégé
78          LDA  DRVRDM,X    ; et positionne en séquence 0
79          EOM
80  WRMODE  MAC              ;Débute l'écriture et écrit
81                           ; l'Accumulateur sur le disque
82          STA  DRVWRM,X    ;Début écriture, charge octet (5)
83          CMP  DRVRD,X     ;Ecrit le 1° octet            (9)
84          EOM
85  WREND   MAC              ;Fin du mode d'écriture
86          LDA  DRVRDM,X
87          LDA  DRVRD,X
88          EOM
89  DELAY6  MAC              ;Délai de 6 cycles
90          NOP
91  DELAY4  MAC              ;Délai de 4 cycles
92          NOP
93  DELAY2  MAC              ;Délai de 2 cycles
94          NOP
95          EOM
96  DELAY3  MAC              ;Délai de 3 cycles
97          BIT  $00
98          EOM
99  DELAY7  MAC              ;Délai de 7 cycles
100         PHA
101         PLA
102         EOM
103 DELAY12A MAC             ;Délai de 12 cycles
104         JSR  RTS1        ;Saut vers un RTS
105         EOM
106 DELAY12B MAC             ;Délai de 12 cycles (identique)
107         JSR  RTS2        ;Saut vers un RTS
108         EOM
109         FIN              ;Fin des macros
110         PUT  FORMAT.SCE1
111         PUT  FORMAT.SCE2
```

Source 'FORMAT.SCE1'
Assembleur ProCode

```
 1  *_____________________________________
 2  *_____________________________________*
 3  *                                     *
 4  *                                     *
 5  *            Début du code            *
 6  *                                     *
 7  *  1ère partie:  Programme Principal  *
 8  *                                     *
 9  *_____________________________________*
10  *_____________________________________*
11
12          PHP                 ;Préserve le contexte
13          SEI                 ; et interdit les interruptions
14          JSR  DEBUT
15          PLP                 ;Restitue le contexte
16          CMP  #$00
17          BNE  ERREUR
18          CLC                 ;Pas d'erreur: Retour à l'envoyeur
19          RTS                 ; avec A = 0 et Carry clear
20
21  * Gestion des erreurs: Conversion en erreurs ProDOS
22  * ---------------------------------------------------
23  *  Entrée: Acc. = n° erreur 1 à 4
24  *  Sortie: Carry set
25  *          Acc. = Erreur ProDOS -> Erreur d'E/S    = $27
26  *                                -> Disque protégé = $2B
27  *                                -> Trop lent      = $33
28  *                                -> Trop rapide    = $34
29
30  ERREUR  CMP  #$02
31          BNE  $autres
32          LDA  #$2B           ;Disque protégé en écriture
33          JMP  $fin
34
35  $autres CMP  #$01
36          BNE  $speed
37          LDA  #$27           ;Erreur d'E/S
38          JMP  $fin
39
40  $speed  CLC
41          ADC  #$30           ;Autres erreurs (vitesse DISK II)
42
43  $fin    SEC                 ;Retour à l'envoyeur avec le code
44          RTS                 ; d'erreur dans A et le carry set
45
46
47  ***********************************************************
48  *                                                         *
49  * Positionnement du bras du lecteur par appel de ARMOVE   *
50  *                                                         *
51  ***********************************************************
52
53  * entrée: A = piste destination
54  *         X = numéro du slot x $10
55  *         PISTACTU = piste actuelle
56  * Sous-programme appelé:
57  *      MOVARM = Positionne le bras suivant la valeur de A
58  * Variable local: PISTDEST
59
60  POSTRACK ASL                ; Piste destination x 2
61          ASL  PISTACTU       ; Piste actuelle x 2
62          STA  PISTDEST       ; A -> Piste destination x 2
63          TXA                 ;Récupère n° du slot x $10
64          LSR
65          LSR
66          LSR
67          LSR
68          TAY                 ;n° du slot en Y
69          LDA  PISTDEST
70          JSR  ARMOVE         ;Positionnement
71          LSR  PISTACTU       ;Récupère piste actuelle
72          RTS
73
74  ****************************
75  *                          *
76  *  formatage d'un disque   *
77  *                          *
78  ****************************
79
80  * Entrée: Interruptions inhibées
81  *  registre A = n° lecteur = n° slot x $10 + $80 si drive 2
82  *          VOLUME  ($DE)  = n° du volume formaté
83  * Sortie: Carry indifférent (positionné par le prog princ)
84  *  registre A = n° de l'erreur de 0 à 4 (décodé par ERREUR)
85
86  DEBUT   TAX
87          AND  #$70           ;Isole le n° du slot
88          STA  SLOT
89          TXA                 ;Récupère n° du slot et du drive
90          LDX  SLOT           ;n° du slot x $10 en X
91          ROL
92          LDA  #$00
93          ROL                 ;Isole n° du drive
94          BNE  $drive2
95          LDA  DRVSL1,X       ;Sélectionne le drive 1
96          JMP  $drive1
97  $drive2 LDA  DRVSL2,X       ;Sélectionne le drive 2
98  $drive1 LDA  DRVON,X        ;Mise en marche du lecteur 1 ou 2
99          LDA  #$D7           ;Initialise l'attente totale = 1 s
100         STA  COMPT1
101         LDA  #$50           ;Position du bras initilisé à $50
102         STA  PISTACTU       ; pour recalibrer celui-ci
103         LDA  #$00           ;Recalibrage sur la piste 0
104         JSR  POSTRACK
105 $attente LDA COMPT1         ;Temporise 1 s depuis le démarrage
106         BEQ  $finatt        ; 41 x 256 boucles
107         JSR  DELAY          ; x 96 microsecondes = 1 seconde
108         JMP  $attente
109 $finatt LDA  VOLUME         ;Passage du paramètre
110         STA  VOL.NB         ; du numéro de volume
111         LDA  #$AA
112         STA  CST.AA
113         LDA  GAPMAX         ; On commence au dessus de la
114         CLC                 ; limite pour pouvoir tester si
115         ADC  #$02           ; le disque n'est pas trop lent
116         STA  GAP3
117         LDA  #$00           ;Première piste à formater
118         STA  PIST.NB        ; = piste 0
119
120 $piste  LDA  PIST.NB        ;Positionnement sur la
121         LDX  SLOT           ; piste à formater
122         JSR  POSTRACK
123         LDX  SLOT
124         WRSTATUS            ;Test si protégé en écriture
125         TAY                 ;Préserve le résultat du test
126         WREND               ;Repositionne les switchs off
127         TYA                 ;Récupère le résultat du test
128         BPL  $okwrite       ;Si non protégée en écriture
129         LDA  #$02           ; sinon erreur 2 = DISQUE PROTEGE
130         JMP  FIN            ; et fin
131 $okwrite JSR WRITRACK       ;formate une piste puis la teste
132         BCC  $okform        ;Si pas d'erreur
133         LDA  #$01           ;Erreur 1 par défaut = ERREUR E/S
134         LDY  GAP3           ;Teste la largeur des gaps
135         CPY  GAPMIN         ;Gap 3 trop court ?
136         BCS  $jmpfin        ;Non: erreur non identifiée et fin
137         LDA  #$04           ; sinon erreur DISK II TROP RAPIDE
138 $jmpfin JMP  FIN            ; et fin
139 $okform LDY  GAP3           ;Pas d'erreur au formatage
140         CPY  GAPMIN         ; mais teste la largeur des gaps
141         BCS  $okspd1        ;Gap 3 trop court ?
142         LDA  #$04           ;Oui: erreur DISK II TROP RAPIDE
143         JMP  FIN            ; et fin
144 $okspd1 CPY  GAPMAX
145         BCC  $okspd2        ;Gap 3 trop long ?
146         LDA  #$03           ;Oui: erreur DISK II TROP LENT
147         JMP  FIN            ; et fin
148 $okspd2 LDA  TESTMAX        ;Ultime test de la piste:
149         STA  TESTNB         ; 16 essais au maximum
150 $test   DEC  TESTNB         ;Décompte le nombre des tests
151         BNE  $encore        ;Encore un test ?
152         LDA  #$01           ; sinon erreur 1 = ERREUR E/S
153         JMP  FIN            ; et fin
154 $encore LDX  SLOT
155         JSR  READRES        ;Tente de lire le secteur zéro
156         BCS  $test          ;Lecture réussie ?
157         LDA  SECTEUR        ;n° du secteur lu
158         BNE  $test          ;Secteur zéro ?
159         LDX  SLOT
160         JSR  READATA        ;puis ses données
161         BCS  $test          ;Lecture réussie ?
162         INC  PIST.NB        ;Test fini on passe à
163         LDA  PIST.NB        ;la piste suivante
164         CMP  #$23           ;Piste 35 (dernière piste) ?
165         BCC  $piste         ;sinon encore une piste
166
167         LDA  #$00           ;Erreur 0 = pas d'erreur
168 FIN     PHA                 ;Préserve le numéro de l'erreur
```

```
169          LDX  SLOT
170          LDA  DRVOFF,X    ;Arrêt du lecteur
171          LDA  #$00        ;Positionne le bras à la piste 0
172          JSR  POSTRACK
173          PLA              ;Récupère le numéro de l'erreur
174          RTS              ;Retour au programme principal
```

Source 'FORMAT.SCE2'

Assembleur ProCode

```
 0
 1  *_____________________________
 2  *_____________________________*
 3  *                             *
 4  *                             *
 5  *  2ème partie: sous-programmes *
 6  *                             *
 7  *_____________________________*
 8  *_____________________________*
 9
10
11  ********************************
12  *                              *
13  *   Test du champ des données  *
14  *                              *
15  ********************************
16
17  * Teste si chacune des 342 ($156) données est nulle ($96)
18  * Entrée:  X = n° de slot x 16
19  *          Read mode ( DRVRD & DRVRDM )
20  * Sortie:   Erreur        => Carry set
21  *           Pas d'erreur => Carry clear
22  * Registres: A = $AA, Y = $00, X = inchangé
23  * Variable locale: COMPTDAT = Compteur des données lues
24
25  READATA  LDY  #$20
26  $posit   DEY
27           BEQ  IZNOGOUD    ;32 tentatives (synchro. bytes) ?
28           READ             ;Lit un octet
29  $reposit EOR  #$D5        ;Début du prologue ?
30           BNE  $posit
31           NOP
32           READ             ;Lit un octet
33           CMP  #$AA        ; Suite du prologue ?
34           BNE  $reposit
35           LDY  #$56        ;Pour après (342 octets)
36           READ             ;Lit un octet
37           CMP  #$AD        ; FIN du prologue des données
38           BNE  $reposit    ; Prologue $D5/$AA/$AD ?
39           LDA  #$00
40  $data    DEY              ;Lecture de 86 données
41           STY  COMPTDAT
42           READ             ;Lit un octet
43           CMP  #$96        ; = donnée nulle
44           BNE  IZNOGOUD    ;Toutes les données nulles ?
45           LDY  COMPTDAT
46           BNE  $data
47  $datas   STY  COMPTDAT    ;Lecture de 256 données
48           READ             ;Lit un octet
49           CMP  #$96        ; = donnée nulle
50           BNE  IZNOGOUD    ;Toutes les données nulles ?
51           LDY  COMPTDAT
52           INY
53           BNE  $datas
54           READ             ;Lit un octet
55           CMP  #$96        ; doit être nulle
56           BNE  IZNOGOUD    ;Checksum correct ?
57           READ             ;Lit un octet
58           CMP  #$DE        ;Début de l'épilogue ?
59           BNE  IZNOGOUD
60           NOP
61           READ             ;Lit un octet
62           CMP  #$AA        ; suite de l'épilogue
63           BEQ  IZGOUD      ;Epilogue = $DE/$AA(/$EB) ?
64  IZNOGOUD SEC              ;Ce n'est pas bon (traduction)
65           RTS

68  ********************************
69  *                              *
70  *   Lecture du champ d'adresse *
71  *                              *
72  ********************************
73
74  * Lecture des n° de volume, piste et secteur
75  * Entrée:  X = n° de slot x 16
76  *          Read mode ( DRVRD & DRVRDM )
77  * Sortie: Erreur        => Carry set
78  *         Pas d'erreur => Carry clear
79  *         TABLADR+3 = n° volume
80  *         TABLADR+2 = n° piste
81  *         TABLADR+1 = n° secteur
82  *         TABLADR   = checksum
83  * Registres: A = $AA, Y = $00, X = inchangé
84  * Variables locales:  COMPTEUR = Compteur d'essai
85  *                     TAMPON1  = Calcul du checksum
86  *                     TAMPON2  = Décodage 4 et 4
87
88  READRES  LDY  #$FC        ;Init. du compteur
89           STY  COMPTEUR
90  $posit   INY               ; 256 tentatives
91           BNE  $encore
92           INC  COMPTEUR     ; x 3 + 4
93           BEQ  IZNOGOUD     ; = 772 tentatives
94  $encore  READ              ;Lit un octet
95  $reposit CMP  #$D5        ; Début du prologue ?
96           BNE  $posit      ; sinon recommence
97           NOP
98           READ             ;Lit un octet
99           CMP  #$AA        ; suite du prologue ?
100          BNE  $reposit
101          LDY  #$03        ;Pour après: VOL/PIST/SECT/CHCKSM
102          READ             ;Lit un octet
103          CMP  #$96        ; fin du prologue
104          BNE  $reposit    ; Prologue = $D5/$AA/$96 ?
105          LDA  #$00        ;Init. pour le checksum
106  $décode STA  TAMPON1     ;Checksum intermédiaire
107          READ             ;Lit un octet
108          ROL              ; x 2 (codage 4 et 4)
109          STA  TAMPON2
110          READ             ;Lit un octet
111          AND  TAMPON2     ;Recompose 4 et 4
112          STA  TABLADR,Y   ;Stocké en ordre décroissant
113          EOR  TAMPON1     ;Calcule la checksum
114          DEY
115          BPL  $décode     ;suivant Vol/Pist/Sect/Chcksm
116          TAY              ; checksum EOR checksum
117          BNE  IZNOGOUD    ; Checksum différent ?
118          READ             ;Lit un octet
119          CMP  #$DE        ;Début de l'épilogue ?
120          BNE  IZNOGOUD
121          NOP
122          READ             ;Lit un octet
123          CMP  #$AA        ; suite de l'épilogue
124          BNE  IZNOGOUD    ; Epilogue = $DE/$AA(/$EB) ?
125  IZGOUD  CLC              ;C'est tout bon ($EB non testé!)
126          RTS
127
128  ******************************************************
129  *                                                    *
130  *   Positionnement du bras sur la piste destination  *
131  *                                                    *
132  ******************************************************
133
134  * Sous programme: DELAY = Temporise suivant la valeur de A
135  * Entrée: registre A = piste destination x 2
136  *         registre X = numéro du slot x $10
137  *         PISTACTU = piste actuelle x 2
138  *         DELAYON et DELAYOFF = tables de temporisation
139  * Sortie: registre X = inchangé
140  * Variables: SLOTARM,PISTDEST,PISTACTU,PISTPREC,PISTDIFF
141  *
142  *  Commutations des switches (Départ sur une piste paire):
143  * Ordre croissant  : 3 0 5 2 4          (1 piste)
144  *              ou  : 3 0 5 2   7 4 1 6 0  (2 pistes) etc...
145  * Ordre décroissant: 7 0 5 6 4          (1 piste)
146  *              ou  : 7 0 5 6   3 4 1 2 0  (2 pistes) etc...
147
148  ARMOVE   STX  SLOTARM
149           STA  PISTDEST    ;Piste destination x 2
150           CMP  PISTACTU    ;Piste actuelle x 2
151           BEQ  $nomove     ;Si égalité, alors fin anticipé
152           LDA  #$00
153           STA  PISTDIFF    ;Nombre de pistes déjà parcourues
154  $boucle  LDA  PISTACTU
155           STA  PISTPREC
156           SEC              ;Piste actuelle - destination x 2
157           SBC  PISTDEST    ; = nombre de pistes à parcourir
158           BEQ  $movend     ;Si = 0 alors fin du mouvement
```

```
159          BCS  $arriere    ;Si actuelle > destination: recule
160          EOR  #$FF        ;Complément à 1: valeur absolue-1
161          INC  PISTACTU    ;Piste suivante en avant
162          BCC  $avant
163 $arriere ADC  #$FE        ;Valeur absolue-1
164          DEC  PISTACTU    ;Piste suivante en arrière
165 $avant   CMP  PISTDIFF    ;Compare pistes à parcourir et
166          BCC  $minimum    ; parcourues. Prend le minimum
167          LDA  PISTDIFF
168 $minimum CMP  #$0C        ;Vitesse limitée au delà de $0C
169          BCS  $limite     ; si égale DELAY inchangé
170          TAY              ; sinon nouveau pointeur
171 $limite  SEC
172          JSR  $on         ;Commute pour la piste suivante
173          LDA  DELAYON,Y   ;Temporise en fonction du
174          JSR  DELAY       ; nombre de pistes parcourues
175          LDA  PISTPREC    ;Piste précedente
176          CLC              ;Décommute celle-ci avec
177          JSR  $off        ; un coup de retard
178          LDA  DELAYOFF,Y  ;Nouvelle temporisation
179          JSR  DELAY
180          INC  PISTDIFF    ;Incrémente pistes parcourues
181          BNE  $boucle     ;Boucle toujours
182 $movend  JSR  DELAY       ;Temporise et décommute
193          CLC              ; la dernière piste courante
184 $on      LDA  PISTACTU
185 $off     AND  #$03        ;Piste modulo 4 ( 4 switches )
186          ROL              ;On ou off suivant la valeur de C
187          ORA  SLOTARM     ;Calcul du switch suivant le slot
188          TAX
189          LDA  DRVS0,X     ;Commute sur la bonne piste
190          LDX  SLOTARM
191 $nomove  RTS
192
193
194 ***************************************************
195 *                                                 *
196 *  Sous-programme d'écriture du champ de données  *
197 *                                                 *
198 ***************************************************
199
200 * Entrée: Registre X  = n° du slot x 16
201 * Sous-programmes appelés: RTS2   = Délai  (DELAY12B)
202 *                          WDATA9 = Ecriture après 9 cycles
203 *                          WDATA7 = Ecriture après 7 cycles
204
205 WRITDATA DELAY12B
206          WRSTATUS         ;Test et initialise l'écriture
207          LDA  #$FF        ;Octets de synchronisation
208          WRMODE           ;Début et écrit le 1[ octet   (9)
209          DELAY7           ;                             (16)
210          DELAY2           ;                             (18)
211          LDY  #$04        ;Encore 4 octets du gap 2     (20)
212 $synchro DELAY7           ;                             (27) (27)
213          JSR  WDATA7      ;27 + 6 + 7 = >40< (15) (15)
214          DEY              ;                  (17) (17)
215          BNE  $synchro    ;                  (19) (20)
216          LDA  #$D5        ;Prologue données  (21)
217          JSR  WDATA9      ; 21 + 6 + 9 = >36<   (15)
218          LDA  #$AA        ;                     (17)
218          JSR  WDATA9      ; 17 + 6 + 9 = >32< cycles
220          LDA  #$AD
221          JSR  WDATA9      ;       (idem)                (15)
222          LDY  #$56        ;  $156 zéros à écrire        (17)
223          DELAY6           ;                             (23)
224          BNE  $premier    ;                             (26)
225 $zéro    DELAY12B         ;Ecriture de $56 zéros  (26)  xx
224 $premier DELAY4           ;                       (30)  (30)
227          LDA  #$96        ;Zéro en code 6 et 2    >32< >32<
228          WRITE            ;Ecrit un octet      (9)  (9)
229          DEY              ;                    (11) (11)
230          BNE  $zéro       ;                    (13) (14)
231          DELAY3           ;                    (16)
232          DELAY2           ;                    (18)
233 $zéros   DELAY12B         ; $100 zéros  (30) (30)
234          LDA  #$96        ;             >32< >32<
235          WRITE            ;Ecrit   (9)   (9)
236          LDA  #$96        ;        (11)  (11)
237          DELAY2           ;        (13)  (13)
238          INY              ;        (15)  (15)
239          BNE  $zéros      ;        (17)  (18)
240          JSR  WDATA9      ;         17 + 6 + 9 = >32<   (15)
241          LDA  #$DE        ;Epilogue des données         (17)
242          JSR  WDATA9      ; 17 + 6 + 9 = >32< cycles
243          LDA  #$AA
244          JSR  WDATA9      ;                (idem)
245          LDA  #$EB        ;Fin épilogue
246          JSR  WDATA9      ;                (idem)
247          LDA  #$FF        ;Octet synchro
248          JSR  WDATA9      ;                (idem)
249          WREND            ;Fin du mode d'écriture
250          RTS
251
252 WDATA9   DELAY2           ;        (2)
253 WDATA7   DELAY7           ; (7)    (9)
254          WRITE            ;Ecrit un octet  (9)
255          RTS              ;                (15)
256
257
258 ***************************************
259 *                                     *
260 *  Ecriture du champ d'adresse        *
261 *                                     *
262 ***************************************
263
264 * Entrée: Registre Y  = GAP3 sauf secteur 0 = 128
265 *         Registre X  = n° du slot x 16
266 *         VOL.NB  = Numéro du volume
267 *         PIST.NB = Numéro de la piste
268 *         SECT.NB = Numéro du secteur
269 *         CST.AA  = $AA (constante)
270 * Sortie: registre X = inchangé
271 *         Carry set si disque protégé, clear sinon
272 * Sous-programmes appelés: RTS1 = Délai   (DELAY12A)
273 *                     WADR.4.4 = Ecriture en codage 4 et 4
274 *                     WDATA11  = Ecriture après 11 cycles
275 *                     WDATA9   = Ecriture après 9 cycles
276
277 WRITADR  SEC              ;Par défaut Carry set ( = erreur)
278          WRSTATUS         ;Test et initialise l'écriture
279          BMI  $fin        ;Si protection alors fin
280          LDA  #$FF        ;Octet de synchro.
281          WRMODE           ;Début et 1[ octet   (9)
282          DELAY7           ;                    (16)
283 $synchro DELAY12A         ;Ecrit les octets  (28) (28)
284          DELAY12A         ; de synchro.      >40< >40<
285          WRITE            ;Ecrit un octet          (9)  (9)
286          DELAY2           ;                       (11) (11)
287          DEY              ;                       (13) (13)
288          BNE  $synchro    ;                       (16) (15)
289          LDA  #$D5        ;Prologue du champ d'adresse (17)
290          JSR  WADR9       ; 17 + 6 + 9 = >32<        (15)
291          LDA  #$AA        ;                          (17)
292          JSR  WADR9       ;                   (idem)
293          LDA  #$96
294          JSR  WADR9       ;Fin prologue       (idem)  (15)
295          LDA  VOL.NB      ;                           (18)
296          JSR  WADR.4.4    ; 18 + 6 + 8 = >32< 6 >32< cycles
297          LDA  PIST.NB
298          JSR  WADR.4.4    ;         (idem)
299          LDA  SECT.NB
300          JSR  WADR.4.4    ;         (idem)              (15)
301          LDA  VOL.NB      ;                             (18)
302          EOR  PIST.NB     ;Calcule l'octet de checksum (21)
303          EOR  SECT.NB     ;                             (24)
304          PHA              ;Préserve le résultat         (27)
305          LSR              ;Décalage (codage 4 et 4)     (29)
306          ORA  CST.AA      ;<=> ORA #$AA mais + 1 cycle >32<
307          WRITEBIS         ;Ecrit un octet            (9)
308          PLA              ;                         (13)
309          ORA  #$AA        ;                         (15)
330          JSR  WADR11      ; 15 + 6 + 11 = >32< cycles  (15)
311          LDA  #$DE        ;Inscrit l'épilogue          (17)
312          JSR  WADR9       ;  17 + 6 + 9 = >32< cycles
313          LDA  #$AA        ;
314          JSR  WADR9       ;                 (idem)
315          LDA  #$EB
316          JSR  WADR9       ;Fin épilogue     (idem)
317          CLC              ;Pas d'erreur
318 $fin     WREND            ;Fin du mode écriture
319 RTS1     RTS
320
321 WADR.4.4 PHA              ;Codage 4 et 4              (3)
322          LSR              ;                           (5)
323          ORA  CST.AA      ;<=> ORA #$AA + 1 cycle     (8)
324          WRITE            ;Ecrit le 1[ nibble     (9)
325          PLA              ;                      (13)
326          DELAY6           ;                      (19)
327          ORA  #$AA        ;                      (21)
328 WADR11   DELAY2           ;       (2)            (23)
329 WADR9    DELAY2           ; (2)   (4)            (25)
330          DELAY7           ; (9)  (11)            >32<
```

```
331          WRITE          ;Ecrit un octet  (9)
332          RTS            ;                (15)
333
334          DFB  $A0,$A0,$A0;Résidu inutile
335
336  ****************************************
337  *                                      *
338  *  Temporisation suivant la valeur de A *
339  *                                      *
340  ****************************************
341  *
342  * Entrée: A = Durée de la temporisation
343  * Temporisation = A x 96 cycles (microsecondes)
344  * COMPT0 et COMPT1 sont utilisés pour temporiser 1 sec.
345
346  DELAY   LDX  #$11
347  $2      DEX
348          BNE  $2        ;Boucle 16 fois
349          INC  COMPT0    ;Incrément de COMPT1 toutes
350          BNE  $3        ; les 256 DELAYs = 24,6 ms
351          INC  COMPT1    ;Nombre / 256 des DELAYs toteaux
352  $3      SEC
353          SBC  #$01
354          BNE  DELAY     ;Boucle A x 96 microsecondes
355          RTS
356
357  *  Table des temporisations
358  *  ------------------------
359  *  Délais permettant une accélération et une
360  * décélération progressives du bras du lecteur.
361
362  DELAYON  DFB  $01,$30,$28;Accélération & décélération
363           DFB  $24,$20,$1E
364           DFB  $1D,$1C,$1C;Vitesse stabilisée
365           DFB  $1C,$1C,$1C
366
367  DELAYOFF DFB  $70,$2C,$26;Idem pour les switches off
368           DFB  $22,$1F,$1E
369           DFB  $1D,$1C,$1C
370           DFB  $1C,$1C,$1C
371
372
373  ****************************
374  *                          *
375  *   Ecriture d'une piste   *
374  *                          *
377  ****************************
378
379  * formate une piste et diminue GAP3 en fonction des tests
380  * Entrée: ESSAIMAX = Nombre maximum d'essais par piste = 3
381  *         GAP3     = Nombre de bit de synchronisation
382  *         SLOT     = Numéro du slot
383  *         TESTMAX  = Nombre maximum de test par secteur
384  * Sortie: Erreur       -> carry set
385  *         Pas d'erreur -> carry clear
386  * Sous-programmes appelés: RTS2      = Temporise 12 cycles
387  *                          WRITADR   = Ecriture des adresses
388  *                          WRITDATA  = Ecriture des données
389  * Variables locales:
390  *   ESSAINB = Nombre d'essais de formatage par piste
391  *   TESTNB  = Nombre de tests de lecture
392  *   SECT.NB = Numéro du secteur en cours
393
394  WRITRACK LDA  ESSAIMAX   ;Nombre maximum (3) d'essais
395           STA  ESSAINB    ;de formatage d'une piste
396  reformat LDY  #$80       ;Longueur du gapl = 128
397           LDA  #$00       ;Premier secteur
398           STA  SECT.NB
359           JMP  $gapl
400  $secteur LDY  GAP3       ;Longueur du gap 3
401  $gapl    LDX  SLOT
402           JSR  WRITADR
403           BCC  $noerr
404           JMP  RTS2       ;En cas d'erreur alors fin
405  $noerr   LDX  SLOT
406           JSR  WRITDATA
407           INC  SECT.NB
408           LDA  SECT.NB
409           CMP  #$10
410           BCC  $secteur   ;Ecriture des 16 secteurs ?
411
412           LDY  #$0F       ;Carte des 16 secteurs
413           STY  SECT.NB
414           LDA  TESTMAX    ;Nombre maximum (16)
415           STA  TESTNB     ;de tentatives de lecture
416  $initmap STA  SECTMAP,Y  ;Initialise la carte à $10
417           DEY
418           BPL  $initmap
419           LDA  GAP3
420           SEC
421           SBC  #$05       ;Nombre de sync. bytes (GAP3) - 5
422           TAY             ;Attente proportionnelle au gap:
423  $waitgap DELAY12B        ;   12  =  12
424           DELAY12B        ; + 12  =  24
428           DELAY7          ; +  7  =  31
426           DELAY4          ; +  4  =  35
427           DEY             ; +  2  =  37
428           BNE  $waitgap   ; +  3  =  40 cycles = 1 sync byte
429           LDX  SLOT
430           JSR  READRES    ;Tente de lire le champ d'adresse
431           BCS  $errgap
432           LDA  SECTEUR    ;n° du secteur lu
439           BEQ  $oksect0   ;Secteur zéro ?
434           DEC  GAP3       ;Sinon GAP3 est trop grand
495           LDA  GAP3       ; on le diminue
436           CMP  GAPMIN     ; GAP3 n'est-il pas trop court ?
437           BCS  $errgap
438           SEC             ;Erreur: le disque va trop vite
439           RTS
440
441  $sectest LDX  SLOT
442           JSR  READRES
443           BCS  $secterr
444  $oksect0 LDX  SLOT
445           JSR  READATA    ;Test des données
446           BCS  $secterr
447           LDY  SECTEUR    ;n° du secteur vérifié
448           LDA  SECTMAP,Y  ;Déjà vérifié ?
449           BMI  $secterr   ;Si oui secteur suivant
450           LDA  #$FF       ;Sinon le marque dans la carte
451           STA  SECTMAP,Y
452           DEC  SECT.NB    ;Un secteur de moins à tester
453           BPL  $sectest   ;Les 16 secteurs sont vérifiés ?
454           CLC             ;Oui, alors OK
455           RTS             ; et c'est fini
456
457  $secterr DEC  TESTNB     ; 16 tentatives de lecture
458           BNE  $sectest   ;Encore une ?
459           DEC  ESSAINB    ; 3 reformatages (GAP3 constant)
460           BNE  $errgap    ;On reformate la piste ?
461           SEC             ;Trop c'est trop
462           RTS             ;KO
463
464  $errgap  LDA  TESTMAX
465           ASL
466           STA  TESTNB     ;TESTMAX x 2 = 32
467  $cherche LDX  SLOT
468           JSR  READRES    ;Recherche le dernier secteur
469           BCS  $e radr
470           LDA  SECTEUR    ;n° du secteur
471           CMP  #$0F       ;Dernier secteur ?
472           BEQ  recomm     ;Oui: reformate un peu plus loin
473  $erradr  DEC  TESTNB     ;Sinon: encore un essai ?
474           BNE  $cherche   ;Cherche pendant 2 tours du disque
475           SEC             ;Secteur 15 non trouvé
476  RTS2     RTS             ;KO (et RTS de temporisation)
477
478  recomm   LDX  #$D6       ;Attente égale à 214 bytes
479  $wait    DELAY12B        ;   12 = 12
480           DELAY12B        ; + 12 = 24
481           DELAY3          ; +  3 = 27
482           DEX             ; +  2 = 29
483           BNE  $wait      ; +  3 = 32 cycles = 1 byte
484           JMP  reformat   ;Recommence un formatage
485
486
487  *  Constantes et variables
488  *  -----------------------
489
490  GAPMIN   DFB  $0E        ;Largeur maximum pour un gap 3
491  GAPMAX   DFB  $1B        ;   "    minimum   "   "  "  "
492  ESSAIMAX DFB  $03        ;Nombre max d'essais de formatage
493  TESTMAX  DFB  $10        ;Nombre maxi de tests de lecture
494
495  SLOT     DS   1          ;Numéro du slot du drive x 16
496  PISTACTU DS   1          ;Piste réel x 2 (seuf init.= $50)
497  TESTNB   DS   1          ;Nombre de tests
498  SECTMAP  DS   16         ;Carte des secteurs vérifiés
499  PISTDEST DS   1          ;Piste destination x 2 (init.= 0)
500  SLOTARM  DS   1          ;Numéro du slot pour MOVARM
501  PISTDIFF DS   1          ;Nombre de piste parcourues
502  PISTPREC DS   1          ;Piste précédente à PISTACTU x 2
```

Tant qu'à faire 5 choses à la fois,

Que ceux qui aiment travailler en faisant deux ou trois choses à la fois ne changent rien, au contraire, avec Apple et Jazz ils peuvent faire mieux.

Jazz de Lotus, c'est un programme créé pour Macintosh 512 Ko, équipé d'un lecteur externe qui permet de devenir un parfait jongleur professionnel.

Cinq programmes en concert, c'est-à-dire un tableur, un grapheur, un gestionnaire de fichier, un traitement de texte et un programme de communication réunis en un seul programme. Ou comment être à cinq sur la même souris.

Pouvoir gérer cinq applications à partir d'un seul écran, modifier une donnée dans une application et qu'elle se modifie automatiquement dans les autres, pouvoir sauter d'un graphe à un traitement de texte sans attendre les secondes qui durent une éternité pour changer de programme, c'est bien...

Quand on s'aperçoit que ces cinq programmes sont individuellement excellents, c'est une révolution.

Jazz, c'est l'outil idéal d'un directeur de service.

Prenons un exemple qui exige beaucoup de souplesse, de rapidité, et de doigté, la direction des services secrets :

– Avec un tableur capable d'afficher 8 192 lignes sur 256 colonnes, la gestion devient beaucoup plus facile. Si un fait nouveau apparaît, rien n'est plus facile que d'étudier deux ou trois hypothèses afin de ne pas être pris de court.

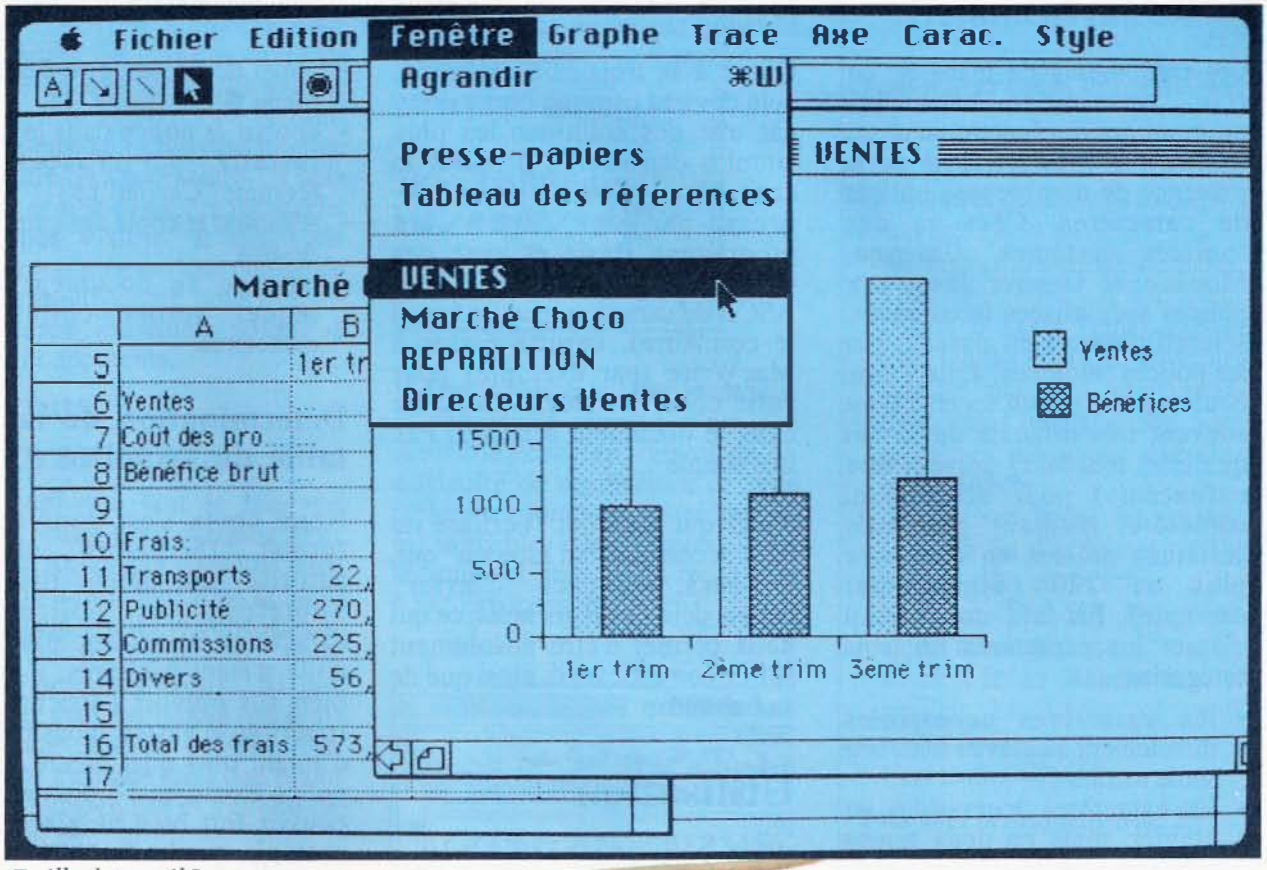
– Quand un projet est fin prêt, il faut bien le présenter à ses supérieurs et parfois même beaucoup plus haut; un grapheur permet de transformer toutes les données numériques obscures en graphiques lumineux.

– Si le projet est accepté, un traitement de texte est nécessaire pour que chaque agent concerné soit au courant dans ses moindres détails. Evidemment quand on travaille dans un tel service il vaut mieux pour Macintosh pouvoir copier, effacer, modifier, ou coller des dessins, à chaque instant.

C.L.M.& BBDO. Macintosh est une marque sous licence chez Apple Computer, Inc.

Apple Computer France. Avenue de l'Océanie

autant les faire en même temps.



Feuille de travail Jazz.

Apple, le logo Apple sont les marques déposées d'Apple Computer, Inc.
Jazz est une marque déposée de Lotus Development Corporation.

Grâce à son gestionnaire de fichier, Jazz fait la fusion automatique entre un mémo et une liste de correspondants à sélectionner selon un critère spécial.
Et avec un modem full duplex 1200 bauds en un instant votre correspondant reçoit toutes les informations précises à l'autre bout du monde.
Avec Jazz l'entreprise est rentable, rondement menée, précise et discrète, inutile d'avoir 35 collaborateurs pour rédiger le projet.

Jazz c'est cinq programmes liés entre eux si parfaitement que l'agent double est enfin enterré, place à l'agent quintuple.

Apple

Z.A. de Courtabœuf. B.P. 131, 91944 Les Ulis Cedex.

AD LITTERAM, POUR TRAVAILLER... A LA LETTRE

Julien Thomas

Les traitements de texte et, en fait, quasiment tous les programmes qui fonctionnent sur le Macintosh bénéficient de la présence de nombreuses polices de caractères. Ceci va des 'polices systèmes' Chicago, Monaco et Genève jusqu'aux polices spécialisées (techniques, scientifiques...) en passant par les polices 'ludiques' style Cairo. Seulement, il y a un revers : il est souvent très difficile de savoir quelle(s) touche(s) doivent être enfoncée(s) pour obtenir tel caractère, d'autant plus que certaines polices en comporte plus de 220 (Times, par exemple). En fait, on pourrait classer les caractères en trois catégories :

- les caractères accessibles directement au clavier avec une seule touche ;
- les caractères accessibles au clavier, mais en deux temps (Option quelque chose plus autre chose) ;
- les caractères absolument inaccessibles soit sur le clavier normal, soit sur le clavier 'étendu' du Macintosh Plus, soit sur les deux.

Pour la première catégorie, la nouvelle version de l'accessoire "clavier" de chez Apple (mais si ! ceux qui ont commencé dans une calculatrice HP, avec des dollars issus de la vente d'un Combi Volkswagen et d'un garage...) fait l'affaire, même si ce n'est pas toujours pratique (on ne peut pas repérer un caractère au premier coup d'œil).

Pour la seconde catégorie, un peu de chance et beaucoup de patience permettent parfois de s'en sortir (qui peut dire instantanément comment on obtient le caractère 'É', sachant que l'on ne l'obtient pas de la même façon sur le Mac et sur le MacPlus ?).

Quant à la troisième catégorie, cela devient presque impossible, car une des solutions les plus simples consisterait à placer le caractère voulu dans le presse-papiers depuis un programme Basic et avec des CHR$(Truc), 'Truc' étant le code ASCII du caractère (encore faut-il le connaître), ensuite passer à MacWrite (par exemple) pour enfin coller le fameux caractère dans le document à traiter. Pas très souple...

Voilà qui justifiait l'écriture du petit accessoire "ad litteram" qui, à Pom's, remplace "Clavier" depuis déjà plusieurs mois, ce qui nous permet d'être absolument sûr de son efficacité, ainsi que de sa fiabilité.

Utilisation

Il n'y a pas grand chose à dire. L'accessoire respectant scrupuleusement l'"interface utilisateur" Macintosh, vous retrouverez ici toute les fonctions qui font le charme de cette machine : Couper/Copier/Coller depuis le menu "Édition" ou au clavier, sélection, sélection étendue, édition, etc. et, si vous disposez d'un MacPlus, déplacements par les touches fléchées.

En fait, si on excepte des points de détail que nous examinerons plus bas, la méthode d'utilisation ce résume à ceci :

- appel de l'accessoire depuis le menu ⌘ (classique !) ;
- choisir la police dans le menu installé par l'accessoire (comme "Clavier") ;
- sélection et copie des caractères désirés ;
- passage au document dans lequel seront collés les caractères.

Détermination de la taille de la police

Notre choix s'est porté sur le format '18 points', ce qui autorise une bonne lisibilité. Toutefois, les polices de caractères n'utilisant pas cette taille n'étant pas rares, il fallait bien sûr prévoir l'affichage en fonction d'autres possibilités, d'autant plus que, même si une police existe en 18 points, vous pouvez fort bien ne placer, par exemple, que les formats 9, 12 et 14 dans le système. Forts de ces constatations qui, vous en conviendrez aisément, ont demandé un effort intellectuel à la limite de l'insoutenable, nous avons employé le 'mécanisme' suivant :

- affichage en 18 points si cette taille existe ;
- sinon, on part à la recherche d'une taille valide inférieure à 18 points : 17, 16, 15... jusqu'à 6 points ;
- si la police en question n'existe pas dans une taille inférieure à 18, on essaie, en désespoir de cause, de trouver un format compris entre 19 et 24, en partant de l'"axe" 18 ;
- si l'on a toujours rien trouvé de valide, il ne nous reste plus qu'une seule solution (mais non pas le suicide... m'enfin !), forcer l'affichage de la police en 24 points, ce qui n'est ni très esthétique ni très rapide, mais on peut se consoler en se disant que ce

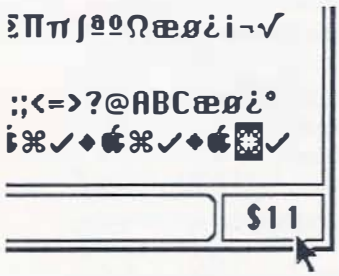
cas de figure n'apparaîtra pratiquement jamais.

Dans le cas, pas fréquent, d'une police existant uniquement en taille 24 points ou plus, ou encore d'une police spécialisée comportant un grand nombre de caractères très larges, il y a de fortes chances pour que certains d'entre-eux se retrouvent hors de la fenêtre de l'accessoire. Si cela arrive, et si le caractère recherché n'est pas visible, il suffit de couper (ou effacer) une chaîne de longueur suffisante pour que les caractères masqués soient à nouveau apparents.

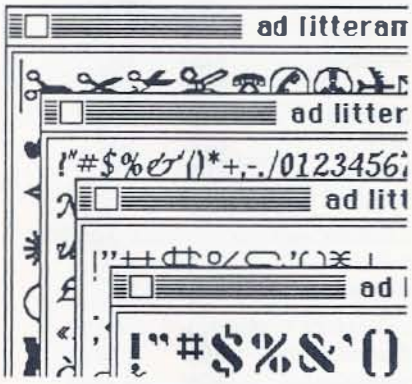
Si vous programmez...

...et quel que soit le langage utilisé, vous êtes certainement amené à rechercher dans une table (que l'on a rarement sous la main au moment voulu) le code ASCII de tel ou tel caractère ; nous avons pensé à vous. Si un caractère est sélectionné (un seul) son code ASCII en notation décimale s'affiche dans une petite 'case' située en bas et à droite de notre fenêtre :

Si vous cliquez sur ce code, vous obtenez le code ASCII en notation hexadécimale, et ce tant que le bouton de la souris est maintenu enfoncé. Pendant cette période, le curseur devient invisible afin de ne pas géner la lecture du code :



Il faut noter que, sur le dessin ci-dessus, le curseur n'est pas invisible, ceci en raison de la difficulté de représenter ce type d'élément lorsque, justement, il est invisible. Sur ce, nous allons maintenant prendre un repos tout à fait indispensable...



Comment faire ?

Vous n'avez pas la disquette Mac 25 :

Pour générer l'accessoire à partir du source et des fichiers listés plus bas, il faut :

- placer, sur une même disquette, l'éditeur (Edit), l'assembleur (Asm), le 'linker' (Link), Exec et Font-DA Mover (avec un fichier système pas trop volumineux, tous ceci trouve place sans problème, même sur une disquette 400Ko) ;
- depuis Exec, lancer 'ad litteram. Job'. En fin d'exécution, et s'il n'y a pas eu d'erreur d'assemblage, Font-DA Mover est appelé automatiquement ;
- ouvrir le fichier 'Accessorium' et installer l'accessoire où vous voulez ;
- c'est tout...

Vous avez la disquette Mac 25 :

- utilisez Font-DA Mover pour installer l'accessoire sur les disquettes de votre choix (il se trouve à la fois dans le système et dans un fichier baptisé 'Accessorium' ;
- c'est tout...

Fichier Exec 'ad litteram.Job'

```
Asm  ad litteram.Asm  Exec          Edit
Link ad litteram.Link Font-DA Mover Edit
```

Fichier 'ad litteram.Link'

```
]
/Resources
ad litteram
/output Accessorium
/Type 'DFIL' 'DMOV'
$
```

Source 'ad litteram.Asm'

Note : le caractère 'ʃ' indique la continuité d'une ligne (ne pas l'insérer dans le code !).

```
RESOURCE 'DRVR' 30 'ad litteram'

; Routines utilisées :

.Trap   _AddResMenu     $A94D
.Trap   _BeginUpdate    $A922
.Trap   _CheckItem      $A945
.Trap   _DeleteMenu     $A936
.Trap   _DisposMenu     $A932
.Trap   _DisposWindow   $A914
.Trap   _DrawControls   $A969
.Trap   _DrawMenuBar    $A937
.Trap   _DrawString     $A884
.Trap   _EndUpdate      $A923
.Trap   _EraseRect      $A8A3
.Trap   _FindControl    $A96C
.Trap   _FrameRect      $A8A1
.Trap   _GetCursor      $A9B9
.Trap   _GetFNum        $A900
.Trap   _GetFontInfo    $A88B
.Trap   _GetHandleSize  $A025
.Trap   _GetItem        $A946
.Trap   _GetMouse       $A972
.Trap   _GetPort        $A874
.Trap   _GetScrap       $A9FD
.Trap   _GlobalToLocal  $A871
.Trap   _HideCursor     $A852
.Trap   _HiliteControl  $A95D
.Trap   _HiliteMenu     $A938
.Trap   _HLock          $A029
.Trap   _HUnLock        $A02A
.Trap   _InitCursor     $A850
```

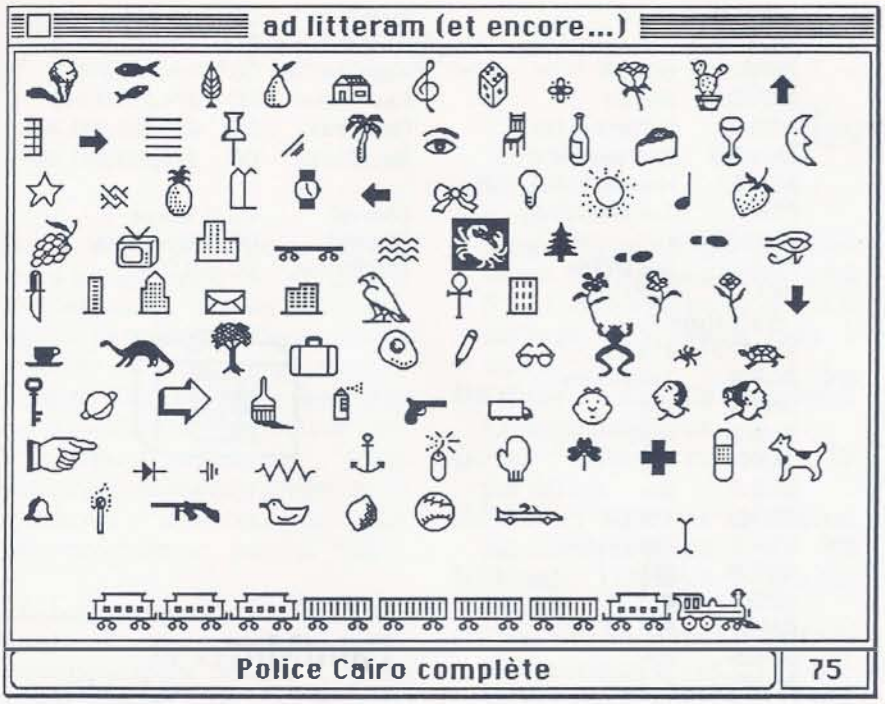
```
.Trap   _InsertMenu      $A935
.Trap   _InvalRect       $A928
.Trap   _MoveTo          $A893
.Trap   _NewControl      $A954
.Trap   _NewMenu         $A931
.Trap   _NewPtr          $A11E
.Trap   _NewWindow       $A913
.Trap   _Pack7           $A9EE
.Trap   _PtInRect        $A8AD
.Trap   _PutScrap        $A9FE
.Trap   _RealFont        $A902
.Trap   _SetCTitle       $A95F
.Trap   _TEActivate      $A9D8
.Trap   _TECalText       $A9D0
.Trap   _TEClick         $A9D4
.Trap   _TECopy          $A9D5
.Trap   _TECut           $A9D6
.Trap   _TEDeactivate    $A9D9
.Trap   _TEDelete        $A9D7
.Trap   _TEDispose       $A9CD
.Trap   _TEIdle          $A9DA
.Trap   _TEInsert        $A9DE
.Trap   _TEKey           $A9DC
.Trap   _TENew           $A9D2
.Trap   _TEPaste         $A9DB
.Trap   _TEUpdate        $A9D3
.Trap   _TextFont        $A887
.Trap   _TextSize        $A88A
.Trap   _TrackControl    $A968
.Trap   _SetCursor       $A851
.Trap   _SetPort         $A873
.Trap   _ShowCursor      $A853
.Trap   _StillDown       $A973
.Trap   _StringWidth     $A88C
.Trap   _SysBeep         $A9C8
.Trap   _ValidRect       $A92A
.Trap   _ZeroScrap       $A9FC

; EQUIVALENCES
; Type d'événements
mButDwnEvt      EQU     1
keyDwnEvt       EQU     3
keyUpEvt        EQU     4
autoKeyEvt      EQU     5
updatEvt        EQU     6
activateEvt     EQU     8
; Décalages/enregistrement événement
evtNum          EQU     $0
evtMessage      EQU     $2
evtMouse        EQU     $A
evtMeta         EQU     $E
evtMBut         EQU     $F
; Messages pour le 'driver' (accessoire)
accEvent        EQU     $40
accCursor       EQU     $42
accMenu         EQU     $43
accUndo         EQU     $44
accCut          EQU     $46
accCopy         EQU     $47
accPaste        EQU     $48
accClear        EQU     $49
; Bits pour fenêtre activée ou
; désactivée et pour touche 'shift'
activeFlag      EQU     0
shiftKey        EQU     9
; Décalages/enregistrement fenêtre
portRect        EQU     $10
WindowKind      EQU     $6C
WindowSize      EQU     $9C
; Type de fenêtre
noGrowDocProc   EQU     4
; Pour la routine de conversion
numToString     EQU     0
; Décalages/'Device Control Entry'
dCtlWindow      EQU     $1E
dCtlRefNum      EQU     $18
dCtlMenu        EQU     $26
; Pour passage des événements
CSCode          EQU     $1A
CSParam         EQU     $1C
; Décalages/rectangle 'QuickDraw'
top             EQU     0
left            EQU     2
bottom          EQU     4
right           EQU     6
; Adresses pour 'Couper/Coller'
TEScrpLength    EQU     $AB0
TEScrpHandle    EQU     $AB4
Scratch8        EQU     $9FA
; Décalages/enregistrement 'TextEdit'
teLineHite      EQU     $18
teAscent        EQU     $1A
teSelStart      EQU     $20
teSelEnd        EQU     $22
teLength        EQU     $3C
teTextH         EQU     $3E
teFont          EQU     $4A
teSize          EQU     $50
; Décalages/enregistrement 'FontInfo'
ascent          EQU     0
descent         EQU     2
; Curseurs 'système'
iBeamCursor     EQU     1
watchCursor     EQU     4
; Pour instruction '_NewPtr'
clear           EQU     $200

; Equivalences 'maison'
BS              EQU     8
MaxC            EQU     300
HandleCtl       EQU     WindowSize
HandleCtl2      EQU     HandleCtl+4
HandlMenu       EQU     HandleCtl2+4
RefMenu         EQU     HandlMenu+4
TamponCtl       EQU     RefMenu+2
TamponTx        EQU     TamponCtl+128
Article         EQU     TamponTx+240
NPoli           EQU     Article+2
DrapeauMdf      EQU     NPoli+2
DrapeauColle    EQU     DrapeauMdf+1
DrapeauCurs     EQU     DrapeauColle+1
CodeCourant     EQU     DrapeauCurs+1
HandlText       EQU     CodeCourant+1
Info            EQU     HandlText+4
HandleAcc       EQU     Info+8
NombreDeci      EQU     HandleAcc+4
PositionND      EQU     NombreDeci+4
NbrHexa         EQU     PositionND+2
PositionNH      EQU     NbrHexa+4
TailleTampon    EQU     PositionNH+2
Oui             EQU     $100
NbrCaracteres   EQU     240

; En-tête de l'accessoire (voir Pom's 19)
Base
        DC      $400
        DC      0
        DC      $16A
        DC      0
        DC      Ouverture-Base
        DC      Status-Base
        DC      Controle-Base
        DC      Status-Base
        DC      Fermeture-Base
Titre
        DC.B    24,'ad litteram (et ∫
                encore...),0
Titre2
        DC.B    7,17,'Pom''s',17

; OUVERTURE DE L'ACCESSOIRE
; Teste si l'accessoire est déjà ouvert.
; Si oui, retour vers l'application.
; Autrement, on débute l'initialisation
; par la création d'un tampon de
; mémoire non relogeable.
Ouverture
        MOVEM.L D3-D7/A1-A4,-(SP)
        MOVEA.L A1,A4
        TST.L   dCtlWindow(A4)
        BNE     Status4
        SUBQ.L  #4,SP
        MOVE.L  SP,-(SP)
        _GetPort
        MOVE.L  #TailleTampon,D0
        _NewPtr,clear
        TST     D0
        BEQ.S   MemoireOK
        MOVE    #7,-(SP)
        _SysBeep
        BRA     Status2

; Ouverture de la fenêtre
MemoireOK
        MOVEA.L A0,A3
        SUBQ.L  #4,SP
        MOVE.L  A3,-(SP)
        PEA     RectFenetre
        PEA     Titre
        MOVE    #Oui,-(SP)
        MOVE    #noGrowDocProc,∫
                -(SP)
        MOVEQ.L #-1,D0
        MOVE.L  D0,-(SP)
        MOVE    #Oui,-(SP)
        CLR.L   -(SP)
        _NewWindow
        _SetPort
        MOVE.L  A3,dCtlWindow(A4)
        MOVE.L  DctlRefNum(A4),∫
                WindowKind(A3)
```

```
; Création du menu 'Pom's'
        MOVE    dCtlRefNum(A4),D0
        NOT     D0
        LSL     #5,D0
        ORI     #$C000,D0
        SUBQ.L  #4,SP
        MOVE    D0,dCtlMenu(A4)
        MOVE    D0,RefMenu(A3)
        MOVE    D0,-(SP)
        PEA     Titre2
        _NewMenu
        MOVE.L  (SP),HandlMenu(A3)
        MOVE.L  #'FONT',-(SP)
        _AddResMenu

; Création du bouton 'Police .. complète'
        SUBQ.L  #4,SP
        MOVE.L  A3,-(SP)
        PEA     RectBouton
        PEA     TitreBouton
        MOVE    #Oui,-(SP)
        CLR.L   -(SP)
        CLR.L   -(SP)
        CLR.L   -(SP)
        _NewControl
        MOVE.L  (SP)+,HandleCtl(A3)

; Initialisation de la chaîne contenant
; les caractéres de code ASCII 16-255

        LEA     TamponTx(A3),A2
        MOVEQ   #$10,D1
        MOVE    #NbrCaracteres-1,D0
@1      MOVE.B  D1,(A2)+
        ADDQ.B  #1,D1
        DBRA    D0,@1

; Création d'un enregistrement
; 'TextEdit', détermination de la police et
; de la taille de départ, initialisations
; diverses et retour à l'application en
; cours.

        LEA     Chaine1,A2
        LEA     TamponCtl+1(A3),A1
        MOVEQ   #6,D0
@2      MOVE.B  (A2)+,(A1)+
        DBRA    D0,@2
        BSR     NouveauTexte
        BSR     Menu2
        ST      DrapeauMdf(A3)
        BSR     Inserer
        PEA     RectVisTexte
        _ValidRect
        SF      DrapeauCurs(A3)
        MOVE    #$0324,NbrHexa(A3)
        _InitCursor
Status2
        _SetPort

Status4
        MOVE.L  dCtlWindow(A4),A3
        MOVE    RefMenu(A3),∫
                dCtlMenu(A4)
Status3
        MOVEM.L (SP)+,D3-D7/A1-A4
Status
        MOVEQ   #0,D0
        RTS

; FERMETURE DE L'ACCESSOIRE
; Suppression de la fenêtre, du menu,
; de l'enregistrement 'TextEdit'...
Fermeture
        MOVEM.L A3-A4,-(SP)
        MOVEA.L A1,A4
        MOVEA.L dCtlWindow(A4),A3
        MOVE.L  HandlText(A3),-(SP)
        _TeDispose
        MOVE    dCtlMenu(A4),-(SP)
        _DeleteMenu
        MOVE.L  HandlMenu(A3),-(SP)
        _DisposMenu
        _DrawMenuBar
        MOVE.L  A3,-(SP)
        _DisposWindow
        CLR.L   dCtlWindow(A4)
        MOVEA.L A4,A1
        MOVEM.L (SP)+,A3-A4
        BRA     Status

; Routine de contrôle. Aiguille
; l'accessoire en fonction des
; messages passés par le système
Controle
        MOVEM.L D3-D7/A1-A4,-(SP)
        MOVEA.L A1,A4
        MOVEA.L A0,A2
        MOVEA.L dCtlWindow(A4),A3
        MOVE.L  A3,-(SP)
        _SetPort
        MOVE    CSCode(A2),D0
        CMPI    #accEvent,D0
        BEQ.S   Evenement
        CMPI    #accCursor,D0
        BEQ     Curseur
        CMPI    #accMenu,D0
        BEQ     Menu
        CMPI    #accCut,D0
        BEQ     Couper
        CMPI    #accCopy,D0
        BEQ     Copier
        CMPI    #accPaste,D0
        BEQ     Coller
        CMPI    #accClear,D0
        BEQ     Effacer
        BRA     Status3

; Aiguillage des événements
Evenement
        MOVEA.L CSParam(A2),A2
        MOVE    EvtNum(A2),D0
        CMPI    #mButDwnEvl,D0
        BEQ.S   Contenu
        CMPI    #keyDwnEvt,D0
        BEQ     Touche
        CMPI    #autoKeyEvt,D0
        BEQ     Touche
        CMPI    #updatEvt,D0
        BEQ     MiseJour
        CMPI    #activateEvt,D0
        BEQ     Active
        BRA     Status3

; On arrive ici en cas de 'click' dans la
; fenêtre de l'accessoire (action sur la
; zone de texte, le bouton 'Police ...
```

```
; complète', ou encore la petite case
; d'affichage du code ASCII du
; caractére sélectionné)
Contenu
        PEA     EvtMouse(A2)
        _GlobalToLocal
        CLR     -(SP)
        MOVE.L  EvtMouse(A2),-(SP)
        MOVE.L  A3,-(SP)
        PEA     HandleCtl2(A3)
        _FindControl
        TST     (SP)+
        BEQ.S   @1
        CLR     -(SP)
        MOVE.L  HandleCtl2(A3),-(SP)
        MOVE.L  EvtMouse(A2),-(SP)
        CLR.L   -(SP)
        _TrackControl
        TST     (SP)+
        BEQ.S   @1
        PEA     RectVisTexte
        _EraseRect
        MOVE.L  HandlText(A3),-(SP)
        _TEDispose
        BSR     NouveauTexte
        BSR     Inserer
        BRA     Status3
@1      CLR     -(SP)
        MOVE.L  EvtMouse(A2),-(SP)
        PEA     RectVisTexte
        _PtInRect
        TST     (SP)+
        BNE.S   @3
        TST.B   CodeCourant(A3)
        BEQ     Status3
        CLR     -(SP)
        MOVE.L  EvtMouse(A2),-(SP)
        PEA     RectCode
        _PtInRect
        TST     (SP)+
        BEQ     Status3
        PEA     RectCode2
        _EraseRect
        BSR     PoliceChicago
        MOVE    PositionNH(A3),-(SP)
        MOVE    #288,-(SP)
        _Moveto
        PEA     NbrHexa(A3)
        _DrawString
        _HideCursor
@4      CLR     -(SP)
        _StillDown
        TST     (SP)+
        BNE.S   @4
        PEA     RectCode2
        _EraseRect
        BSR     AfficheCode
        _ShowCursor
        BRA     Status3
@3      MOVE.L  evtMouse(A2),-(SP)
        BTST    #shiftKey,∫
                evtMeta(A2)
```

Mac 128
Mac 512
Mac Plus

```
        SNE     D0
        MOVE.B  D0,-(SP)
        MOVE.L  HandlText(A3),-(SP)
        _TEClick
        BSR     DeReference
        MOVE    teSelStart(A0),D0
        MOVE    teSelEnd(A0),D1
        SUB     D0,D1
        SUBQ    #1,D1
        BNE     @2
        MOVEA.L teTextH(A0),A0
        MOVEA.L (A0),A0
        MOVE.B  0(A0,D0),D0
        CMP.B   CodeCourant(A3),D0
        BEQ     Status3
        MOVE.B  D0,CodeCourant(A3)
        ANDI.L  #$FF,D0
        LEA     NombreDeci(A3),A0
        CLR     -(SP)
        _Pack7
        BSR     PoliceChicago
        CLR     -(SP)
        PEA     NombreDeci(A3)
        _StringWidth
        MOVE    #38,D0
        SUB     (SP)+,D0
        LSR     #1,D0
        ADD     #351,D0
        MOVE    D0,PositionND(A3)
        CLR.L   D0
        MOVE.B  CodeCourant(A3),D0
        DIVU    #16,D0
        BSR     CodeHexa
        MOVE.B  D0,NbrHexa+2(A3)
        SWAP    D0
        BSR     CodeHexa
        MOVE.B  D0,NbrHexa+3(A3)
        CLR     -(SP)
        PEA     NbrHexa(A3)
        _StringWidth
        MOVE    #38,D0
        SUB     (SP)+,D0
        LSR     #1,D0
        ADD     #351,D0
        MOVE    D0,PositionNH(A3)
        PEA     RectCode2
        _EraseRect
        BSR     AfficheCode
        BRA     Status3
@2      TST.B   CodeCourant(A3)
        BEQ     Status3
        PEA     RectCode2
        _EraseRect
        SF      CodeCourant(A3)
        BRA     Status3

; Segment utilisé en cas d'action sur le
; clavier. On détermine s'il s'agit d'un
; caractère à insérer dans le texte ou
; d'une commande (couper, copier ou
; coller)
Touche
        BTST    #0,evtMeta(A2)
        BEQ.S   @2
        MOVE    evtMessage+2(A2),∫
                D0
        ANDI    #$DF,D0
        SUBI    #86,D0
        BEQ     Coller
        SUBQ    #2,D0
        BEQ     Couper
        SUBI    #$FFEB,D0
        BEQ     Copier
        BRA     Status3
@2      CMPI.B  #BS,∫
                evtMessage+3(A2)
        BEQ.S   @1
        BSR     DeReference
        CMPI    #MaxC,teLength(A0)
        BLT.S   @1
        MOVE    #7,-(SP)
        _SysBeep
        BRA     Status3
@1      MOVE    evtMessage+2(A2),∫
                -(SP)
        MOVE.L  HandlText(A3),-(SP)
        _TEKey
        BSR     ValideBouton
        BSR     EffaceCode
        BRA     Status3

; Routine de mise à jour
MiseJour
        MOVE.L  A3,-(SP)
        _BeginUpdate
        PEA     RectTexte
        _FrameRect
        PEA     RectCode
        _FrameRect
        TST.B   CodeCourant(A3)
        BEQ.S   @1
        BSR     PoliceChicago
        BSR     AfficheCode
@1      MOVE.L  A3,-(SP)
        _DrawControls
        PEA     RectVisTexte
        _EraseRect
        PEA     portRect(A3)
        MOVE.L  HandlText(A3),-(SP)
        _TEUpdate
        MOVE.L  A3,-(SP)
        _EndUpdate
        BRA     Status3

; Réponse à un message 'Activate' ou
; 'Deactivate' pour rendre active ou
; inactive la fenêtre de l'accessoire
Active
        BTST    #activeFlag,∫
                evtMBut(A2)
        BEQ.S   @1
        MOVE.L  HandlText(A3),-(SP)
        _TEActivate
        SF      DrapeauColle(A3)
        MOVE.L  HandlMenu(A3),-(SP)
        CLR     -(SP)
        _InsertMenu
        _DrawMenuBar
        BRA     Status3
@1      MOVE.L  HandlText(A3),-(SP)
        _TEDeactivate
        MOVE    RefMenu(A3),-(SP)
        _DeleteMenu
        _DrawMenuBar
        BRA     Status3

; Changement de la forme du curseur si
; nécessaire (flèche ou curseur type
; 'texte' en forme de I)
Curseur
Buffer          SET     -4
        LINK    A6,#Buffer
        MOVE.L  HandlText(A3),-(SP)
        _TEIdle
        PEA     Buffer(A6)
        _GetMouse
        CLR     -(SP)
        MOVE.L  Buffer(A6),-(SP)
        PEA     RectVisTexte
        _PtInRect
        TST     (SP)+
        BEQ.S   @1
        TST.B   DrapeauCurs(A3)
        BNE.S   @2
        SUBQ.L  #4,SP
        MOVE    #iBeamCursor,-(SP)
        _GetCursor
        MOVEA.L (SP)+,A0
        MOVE.L  (A0),-(SP)
        _SetCursor
        ST      DrapeauCurs(A3)
        BRA.S   @2
@1      TST.B   DrapeauCurs(A3)
        BEQ.S   @2
        SF      DrapeauCurs(A3)
        _InitCursor
@2      UNLK    A6
        BRA     Status3

; Traitement d'un changement de police
; (menu 'Pom's')
Menu
        MOVE    CSParam+2(A2),D7
        CMP     Article(A3),D7
        BEQ     Status3
        SF      D6
        BRA.S   L1
Menu2
        MOVEQ   #1,D7
        CLR     Article(A3)
        ST      D6
L1      SUBQ    #4,SP
        MOVE    #watchCursor,-(SP)
        _GetCursor
        MOVEA.L (SP)+,A2
        MOVE.L  (A2),-(SP)
        _Setcursor
        BSR     MenuPolices
        MOVE    NPoli(A3),-(SP)
        _TextFont
        MOVE    D7,-(SP)
        _TextSize
        PEA     Info(A3)
        _GetFontInfo
        BSR     DeReference
        MOVE    NPoli(A3),teFont(A0)
        MOVE    D7,teSize(A0)
        MOVE    Info+ascent(A3),D0
        MOVE    D0,teAscent(A0)
```




```
      ADD       Info+descent(A3),D0
      MOVE      D0,teLineHite(A0)
      MOVE.L    HandlText(A3),-(SP)
      _TeCalText
      TST.B     D6
      BEO.S     @1
      RTS
@1    PEA       RectVisTexte
      _InvalRect
      CLR       -(SP)
      _HiliteMenu
      SF        DrapeauCurs(A3)
      _InitCursor
      BRA       Status3

; COUPER
Couper
      BSR       VerifSelection
      BEQ       Status3
      MOVE.L    HandlText(A3),-(SP)
      _TeCut
      BSR       TEToScrap
      BSR       ValideBouton
      BRA       Status3

; COPIER
Copier
      BSR       VerifSelection
      BEQ       Status3
      MOVE.L    HandlText(A3),-(SP)
      _TeCopy
      BSR       TEToScrap
      BRA       Status3

;COLLER
Coller
      TST.B     DrapeauColle(A3)
      BNE.S     @2
      ST        DrapeauColle(A3)
      BSR       TEFromScrap
@2    MOVEA.L   HandlText(A3),A0
      MOVEA.L   (A0),A0
      MOVE      TeScrpLength,D0
      ADD       teLength(A0),D0
      MOVE      teSelEnd(A0),D1
      MOVE      teSelStart(A0),D2
      SUB       D2,D1
      SUB       D1,D0
      CMPI      #MaxC,D0
      BLT.S     @1
      MOVE      #7,-(SP)
      _SysBeep
      BRA       Status3
@1    TST       TeScrpLength
      BEQ       Status3
      MOVE.L    HandlText(A3),-(SP)
      _TePaste
      BSR       ValideBouton
      BRA       Status3

; EFFACER
Effacer
      BSR       VerifSelection
      BEQ       Status3
      MOVE.L    HandlText(A3),-(SP)
      _TEDelete
      BSR       ValideBouton
      BRA       Status3

; Sous-programme pour valider le
; bouton 'Police ... complète'
ValideBouton
      TST.B     DrapeauMdf(A3)
      BNE.S     @1
      MOVE.L    HandleCtl(A3),-(SP)
      CLR       -(SP)
      _HiliteControl
      ST        DrapeauMdf(A3)
@1    RTS

; Sous-programme pour invalider le
; bouton
InvalideBouton
      TST.B     DrapeauMdf(A3)
      BEO.S     @1
      MOVE.L    HandleCtl(A3),-(SP)
      MOVE      #$FF,-(SP)
      _HiliteControl
      SF        DrapeauMdf(A3)
@1    RTS

; Passage du texte du presse-papiers
; 'DeskScrap' au presse-papiers
; 'TextEdit' (voir Pom's 24)
TEFromScrap
      MOVEM.L   A0-A1/D0-D2,-(SP)
      SUBQ      #4,SP
      MOVE.L    teScrpHandle,-(SP)
      MOVE.L    #'TEXT',-(SP)
      PEA       scratch8
      _GetScrap
      MOVE.L    (SP)+,D0
      BPL.S     @1
      MOVEQ     #0,D0
@1    MOVE      D0,TeScrpLength
      MOVEM.L   (SP)+,A0-A1/D0-D2
      RTS

; Passage du texte du presse-papiers
; 'TextEdit' au presse-papiers
; 'DeskScrap' (voir Pom's 24)
TEToScrap
      MOVEM.L   A0-A1/D0-D2,-(SP)
      SUBQ      #4,SP
      _ZeroScrap
      ADDQ      #4,SP
      MOVEA.L   teScrpHandle,A0
      SUBQ      #4,SP
      _GetHandleSize
      MOVE.L    D0,-(SP)
      _HLock
      MOVE.L    #'TEXT',-(SP)
      MOVE.L    (A0),-(SP)
      _PutScrap
      MOVEA.L   teScrpHandle,A0
      _HUnLock
      MOVE.L    (SP)+,D0
      MOVEM.L   (SP)+,A0-A1/D0-D2
      RTS

; Efface le code ASCII
EffaceCode
      TST.B     CodeCourant(A3)
      BEQ.S     @1
      PEA       RectCode2
      _EraseRect
      SF        CodeCourant(A3)
@1    RTS

; Sous-programme pour vérifier si des
; caractères sont sélectionnés
VerifSelection
      BSR.S     DeReference
      MOVE      teSelStart(A0),D0
```

```
        CMP     teSelEnd(A0),D0
        RTS

; 'Déréférence' le 'Handle' de
; l'enregistrement 'TextEdit'
DeReference
        MOVEA.L HandlText(A3),A0
        MOVEA.L (A0),A0
        RTS

; Création d'un nouvel enregistrement
; 'TextEdit'
NouveauTexte
        SUBQ.L  #4,SP
        PEA     RectDestText
        PEA     RectVisTexte
        _TENew
        MOVE.L  (SP)+,HandlText(A3)
        BSR     EffaceCode
        RTS

; Insertion de la chaîne de caractères
; (ASCII 16 à 255)
Inserer
        PEA     TamponTx(A3)
        MOVE.L  #NbrCaracteres,-(SP)
        MOVE.L  HandlText(A3),-(SP)
        _TEInsert
        MOVE.L  HandlText(A3),-(SP)
        _TEActivate
        BSR     InvalideBouton
        RTS

; Mise à jour du menu 'Pom's' et
; changement du texte et du bouton en
; fonction de la police sélectionnée
MenuPolices
Buffer  SET     -100
        LINK    A6,#Buffer
        TST     Article(A3)
        BEQ.S   @1
        MOVE.L  HandlMenu(A3),-(SP)
        MOVE    Article(A3),-(SP)
        CLR     -(SP)
        _CheckItem
@1      MOVE    D7,Article(A3)
        MOVE.L  HandlMenu(A3),-(SP)
        MOVE    D7,-(SP)
        ST      -(SP)
        _CheckItem
        MOVE.L  HandlMenu(A3),-(SP)
        MOVE    D7,-(SP)
        PEA     Buffer(A6)
        _GetItem
        LEA     Buffer(A6),A2
        LEA     TamponCtl+8(A3),A1
        MOVE.B  (A2)+,D0
        EXT     D0
        SUBQ    #1,D0
@2      MOVE.B  (A2)+,(A1)+
        DBRA    D0,@2
        LEA     Chaine2,A2
        MOVEQ   #8,D0
@3      MOVE.B  (A2)+,(A1)+
        DBRA    D0,@3
        MOVEQ   #16,D0
        ADD.B   Buffer(A6),D0
        MOVE.B  D0,TamponCtl(A3)
        MOVE.L  HandleCtl(A3),-(SP)
        PEA     TamponCtl(A3)
        _SetCTitle
        PEA     Buffer(A6)
        PEA     NPoli(A3)
        _GetFNum
        MOVEQ   #18,D7
@4      BSR.S   TailleValide
        BNE.S   @5
        SUBQ    #1,D7
        CMPI    #5,D7
        BNE.S   @4
        MOVE    #19,D7
@6      BSR.S   TailleValide
        BNE.S   @5
        ADDQ    #1,D7
        CMPI    #25,D7
        BNE.S   @6
        MOVEQ   #18,D7
@5      UNLK    A6
        RTS

; Pour déterminer si la police
; sélectionnée existe en taille N (D7)
TailleValide
        SUBQ.L  #2,SP
        MOVE    NPoli(A3),-(SP)
        MOVE    D7,-(SP)
        _RealFont
        TST     (SP)+
        RTS

; Affiche le code ASCII (décimal) du
; caractère sélectionné
AfficheCode
        MOVE    PositionND(A3),-(SP)
        MOVE    #288,-(SP)
        _MoveTo
        PEA     NombreDeci(A3)
        _DrewString
        RTS

; Décimal -> Hexadécimal
CodeHexa
        CMPI.B  #9,D0
        BGT     @1
        ORI.B   #$30,D0
        RTS
@1      ADDI.B  #55,D0
        RTS

; Pour police Chicago 12 points
PoliceChicago
        CLR     -(SP)
        _TextFont
        MOVE    #12,-(SP)
        _TextSize
        RTS

; Constantes
RectFenetre     DC      41,4,336,396
RectTexte       DC      2,2,271,390
RectVisTexte    DC      4,4,269,388
RectDestText    DC      6,6,450,376
RectBouton      DC      274,2,293,348
RectCode        DC      274,350,293,390
RectCode2       DC      275,351,292,389

Chaine1         DC.B 'Police '
Chaine2         DC.B ' complète'
TitreBouton     DC.B 0,' '
```

PaintMover et MacPlus

PaintMover nécessite plus de 128Ko pour fonctionner ; aussi, si vous avez réservé 768 Ko de cache mémoire, le programme se 'plantera', hélas sans message...

Et pourtant, il fonctionne !

B.-P. Eminet et É. Vernier dans un livre consacré au Basic Microsoft sur Macintosh, publié aux éditions McGraw-Hill font référence au programme de J.-L. Bazanegue paru dans Pom's 16 ('personnalisez vos disquettes Macintosh'), programme qui ne fonctionnerait pas...

Doit-on rappeler ici que :

- ce programme était conçu pour le Basic 1.0 ;
- à l'époque, le Basic 2.0 n'était pas diffusé ;
- l'adaptation au Basic 2.0 a été publiée dans le numéro 17 ;
- même ainsi adapté, ce programme a toutes les chances de ne pas fonctionner avec une hypothétique version 4.12 ou 6.0 de ce même Basic.

Moralité, quand on ne sait pas...

Ascenseurs en Basic

Bernard Baz

Nous vous présentons ici un sous-programme à variables locales, donc 'transparent' qui vous permettra d'initialiser, afficher et gérer un ascenseur. Comme il est inutile de présenter cette élément important de l''interface utilisateur', nous passeront immédiatement à la méthode d'utilisation du sous-programme, baptisé "GF".

Le sous-programme "GF", tel qu'il est présenté ici, est essentiellement destiné à gérer une fenêtre de texte. Le programme de démonstration met ceci en évidence en affichant des lignes de texte avec différentes polices et en différentes tailles, les déplacements verticaux de ces lignes suite aux actions sur l'ascenseur étant proportionnels aux caractères utilisés. Vous pourrez toutefois utiliser "GF" pour autre chose que du texte en remplaçant la routine 'Afficher' par la routine de votre choix.

Arguments d'appel de GF

Il s'agit d'une valeur permettant d'indiquer au sous-programme la tâche à accomplir, ceci en fonction des message passés par le système (ON DIALOG) :

1 - **Initialisation**. Prise en compte des paramètres et de la taille de la fenêtre. Cet appel est nécessaire chaque fois que les paramètres ou que la taille de la fenêtre à gérer sont modifiés.
2 - **Rafraîchissement** de la fenêtre.
3 - **Défilement** de la fenêtre sur événement souris.

Paramètres d'initialisation (pour argument = 1)

GFlim : numéro de la dernière ligne affichable. (La numérotation des lignes commence à 0). Une valeur négative indique qu'il ny a rien à afficher.

GFtfont : police de caractères. Chicago par défaut.

GFtsiz : taille des caractères. 12 par défaut.

GFtface : style des caractères. Standard par défaut.

GFtitre$: Titre. Pas de titre par défaut.

GFpas : nombre de lignes (≥0) en recouvrement lors de l'affichage d'écrans successifs. Généralement 0 ou 1. 0 par défaut.

GFx1, GFy1, GFx2, GFy2 : coordonnées (coin supérieur gauche, coin inférieur droit) de la sous-fenêtre gérée par GF. Les informations affichées à l'extérieur de cette sous-fenêtre ne sont pas affectées par le défilement.
Les valeurs incorrectes sont automatiquement ramenées aux valeurs par défaut, soit : 0, 18, largeur de la fenêtre-16, hauteur de la fenêtre.

Variable disponible après chaque appel à GF sur événement souris (argument = 3)

GFmouse : 0 si l'événement a été pris en compte par GF, -1 si l'événement ne concerne pas GF et peut être traité par le programme appelant.

Sous-programme utilisateur appelé par GF : Afficher(x,y)

L'appel de ce sous-programme correspond au traitement d'une action sur l'ascenseur. Il pourrait être remplacé par un autre sous-programme pour, par exemple, faire défiler du graphisme.

x : numéro de la première ligne à afficher (commence à 0).
y : nombre de lignes à afficher (le curseur est positionné par GF)

Remarques : la fenêtre à gérer doit être active et être la fenêtre courante d'affichage. Elle doit être ouverte avant l'appel d'initialisation.

Enfin, pour que la présentation soit plus agréable, utilisez plutôt les types 3 ou 4. Un bouton invisible (N° 255) est déclaré par GF afin d'inhiber le mode "scrolling" automatique.

Toutes les variables de GF sont de type entier implicite (DEFINT a-z).

Si les événements souris sont traités de façon asynchrone (ON MOUSE...) et non par "polling" (IF MOUSE(0)<>0...), ils doivent être inhibés (MOUSE STOP) pendant un rafraîchissement puis autorisés à nouveau (MOUSE ON) lorsque celui-ci est terminé.

Ceci évite le risque d'appeler GF sur événement souris alors qu'il est en train d'effectuer un rafraîchissement.

Au retour de GF 3 le programme peut examiner la variable GFmouse pour savoir si l'événement a été traité par GF (GFmouse=0) ou si l'utilisateur n'avait pas cliqué dans la bande de défilement (GFmouse=-1).

Dans ce dernier cas, le programme appelant peut traiter lui-même l'événement. Les DATA de GF sont lues une fois pour toutes lors du premier appel d'initialisation.

Si le programme appelant a lui-même des DATA à lire, il devra les placer avant ou après celles de GF selon quelles seront lues avant ou après le premier appel à GF(1).

Démonstration du sous-programme 'GF'

***Note** : pour un Mac 128Ko, placez en début de programme : **CLEAR** ,11000,1024*

```
DEFINT A-Z
WINDOW 1,,(2,23)-(508,330),
  3
' Fonctions appelées par GF
DEF FNmin(x,y)=x<=y AND x O
  R y<x AND y
DEF FNmax(x,y)=x>=y AND x O
  R y>x AND y
GFLim=100
GFtfont=3
GFtitre$="Exemple d'utilisat
  ion"
GFy2=WINDOW(3)-10

GF 1:'Initialisation
TEXTMODE 1:MENU 6,0,1,"Car
  actères":MENU 6,1,1,"Chica
  go":MENU 6,2,1,"New York":
  MENU 6,3,2,"Geneva":MENU
  6,4,1,"Monaco":MENU 7,0,1,
  "Style":MENU 7,1,2,"Standa
  rd":MENU 7,2,1,"Gras":MEN
  U 7,3,1,"Italique"
MENU 7,4,1,"Souligné":MENU
  7,5,1,"Relief":MENU 7,6,1,
  "Ombre":MENU 7,7,0,"-":ME
  NU 7,8,1,"9 Points":MENU
  7,9,1,"10 Points":MENU 7,1
  0,2,"12 Points":ON MOUSE
  GOSUB EvtSou:MOUSE ON
Attente:
WHILE 1
IF DIALOG(0)=5 THEN GOSUB
   MajFen
GOSUB EvtMen
WEND
MajFen:
MOUSE STOP:TEXTSIZE 9:TEX
  TFONT 3:TEXTFACE 0:MOVE
  TO WINDOW(2)-160,GFy2+5:P
  RINT"GF V1.8 © Bernard BA
  Z, 1986"
GF 2: 'Rafraîchissement
MOUSE ON:RETURN
EvtSou:
GF 3:'Gestion souris
IF GFmouse THEN BEEP
RETURN
EvtMen:
w=MENU(0):IF w>0 THEN MENU
   w,0,1:ON w-5 GOSUB Car,S
  tyle
IF aff THEN aff=0:GF 1:CLS:
  GOSUB MajFen
RETURN
Car:
IF MENU(1)=1 THEN w=0 ELSE
   w=MENU(1)
IF w<>GFtfont THEN si=1
WHILE si:si=0
MENU 6,FNmax(GFtfont,1),1:M
  ENU 6,MENU(1),2:GFtfont=w
  :aff=1
WEND:RETURN
Style:
IF MENU(1)>7 THEN GOSUB T
  aille:RETURN
IF MENU(1)=1 AND GFtface>0
  THEN si=1
WHILE si:si=0
FOR i=2 TO 6:MENU 7,i,1:NE
  XT:MENU 7,1,2:GFtface=0:a
  ff=1
WEND:IF MENU(1)>1 THEN si=
  1
WHILE si:si=0
w=GFtface XOR 2^(MENU(1)-2)
  :MENU 7,MENU(1),1-(w>GFtf
  ace):GFtface=w:IF GFtface=
  0 THEN MENU 7,1,2 ELSE M
  ENU 7,1,1
aff=1
WEND:RETURN
Taille:
IF MENU(1)<10 THEN w=MENU(
  1)+1 ELSE w=12
IF w<> GFtsiz THEN si=1
WHILE si: si=0:FOR i=8 TO 1
  0:MENU 7,i,1:NEXT:MENU 7
  ,MENU(1),2:GFtsiz=w:aff=1:
  WEND:RETURN
SUB Afficher(li,nli) STATIC
FOR i=0 TO nli-1:PRINT"Lign
  e";li+i:NEXT
END SUB
'GF V1.8 © Bernard BAZ, 1986
SUB GF(w) STATIC
ON w GOSUB GFinit,GFmaj,GFs
  ou
EXIT SUB
GFinit:
WHILE NOT init:init=-1
SHARED GFlim,GFtfont,GFtsiz
  ,GFtface,GFtitre$,GFpas,GF
  x1,GFx2,GFy1,GFy2,GFmouse
DIM fls(18),fli(18),flsn(18)
  ,flin(18),bde(4),asr(4),gr
  is(4),lin(4)
FOR i=1 TO 18:READ fls(i):N
  EXT
fli(1)=16:fli(2)=16:FOR i=3
  TO 18:fli(i)=fls(21-i):NE
  XT
FOR i=1 TO 14:READ flsn(i):
  NEXT
flin(1)=16:flin(2)=16:FOR i=
  3 TO 18:flin(i)=flsn(21-i)
  :NEXT
FOR i=1 TO 4:gris(i)=&h1144
  :lin(i)=255:NEXT
BUTTON 255,0,"",(0,0)-(0,0)
WEND
lfen=WINDOW(2):hfen=WINDOW
  (3)
xfl=lfen-15:yfli=hfen-15
bde(1)=33:bde(2)=lfen-14:bde
  (3)=hfen-15:bde(4)=lfen
asr(1)=33:asr(2)=lfen-14:asr
  (3)=49:asr(4)=lfen
TEXTFONT 0:TEXTSIZE 12
tit$=GFtitre$
WHILE WIDTH(tit$)>lfen-13:t
  it$=LEFT$(tit$,LEN(tit$)-
  1):WEND
xtit=(lfen-1-WIDTH(tit$))/2
  :tsiz=GFtsiz OR 12 AND GF
  tsiz=0:lim=GFlim:li1=0
TEXTFONT GFtfont:TEXTSIZE
  tsiz:TEXTFACE GFtface:LO
  CATE 1,1:x1i=WINDOW(4):y1
  i=WINDOW(5):LOCATE 2,1:h
  1i=WINDOW(5)-y1i
IF GFx1<0 OR GFx1>=lfen-16
  THEN GFx1=0
IF GFx2<=GFx1 OR GFx2>lfen-1
  6 THEN GFx2=lfen-16
IF GFy1<18 OR GFy1>hfen-hli
  THEN GFy1=18
IF GFy2<GFy1+hli OR GFy2>hfe
  n-1 THEN GFy2=hfen-1
xli=GFx1+xli:yli=GFy1+yli:li
  pg=FNmax(1,(GFy2-GFy1+1)\h
  li):pas=FNmin(FNmax(0,GFpa
  s),lipg-1):li1m=FNmax(0,li
  m-lipg+1):x1rol=GFx1:x2rol
  =GFx2:y1rol=GFy1:y2rol=y1r
  ol+hli*lipg-1:RETURN
GFmaj:
TEXTFONT 0:TEXTSIZE 12:TE
  XTFACE 0:PENPAT VARPTR(
  lin(1)):PENSIZE 1,12:MOV
  ETO 1,2:LINETO xtit-6,2:M
  OVE 7,11:PRINT tit$;:MOV
  E 6,-11: LINETO lfen-2,2:
  PENNORMAL
LINE(0,17)-(lfen,17):LINE(x
  fl,17)-(xfl,hfen):PUT (xfl
  ,17),fls(1),PSET:PUT(xfl,
  yfli),fli(1),PSET:IF lim>
  =lipg THEN si=1
WHILE si:si=0:FILLRECT VA
  RPTR(bde(1)),VARPTR(gris(
  1)):ERASERECT VARPTR(asr
  (1)):FRAMERECT VARPTR(as
  r(1)):WEND
TEXTFONT GFtfont:TEXTSIZE
  tsiz:TEXTFACE GFtface:MO
  VETO xli,yli:IF lim>=0 TH
  EN Afficher li1,FNmin(lim,
  lipg-1)+1
RETURN
```

```
GFsou:
w=MOUSE(0):si=MOUSE(3)>=xfl
   AND MOUSE(4)>=17:GFmouse
  =NOT si
WHILE si AND lim>=lipg:si=0
zon=1-(MOUSE(4)>32)-(MOUSE(
  4)>=asr(1))-(MOUSE(4)>=as
  r(3))-(MOUSE(4)>=yfli)
ON zon GOSUB Zon1,Zon2,Zon3
  ,Zon4,Zon5
WEND
RETURN
Zon1:
PUT(xfl,17),flsn(1):act=1
WHILE act:IF lil>0 THEN li
  l=lil-1:GOSUB aff
act=MOUSE(0)<0 AND MOUSE(6
  )<=32
WEND
PUT(xfl,17),flsn(1):RETURN
Zon2:
mou=MOUSE(4):act=1
WHILE act:IF lil>0 THEN li
  l=FNmax(lil-lipg+pas,0):GO
  SUB aff
act=MOUSE(0)<0 AND MOUSE(4
  )=mou AND MOUSE(6)<asr(1)
WEND:RETURN
Zon3:
PENPAT VARPTR(gris(1)):PEN
  MODE 14:FRAMERECT VARPT
  R(asr(1)):yl=asr(1):act=1:
  si=1
WHILE MOUSE(0)<0 OR si:si=
  0
y2=FNmax(FNmin(yl+MOUSE(6)-
  MOUSE(4),hfen-31),33)
IF y2<>asr(1) AND act THEN
  si=1 ELSE asr(1)=y2: asr(3
  )=y2+16
WHILE si:si=0:FRAMERECT VA
  RPTR(asr(1)):asr(1)=y2: as
  r(3)=y2+16:FRAMERECT VAR
  PTR(asr(1))
WEND
IF ABS(lfen-8-MOUSE(5))>30
   AND act THEN act=0:FRAM
  ERECT VARPTR(asr(1))
IF ABS(lfen-8-MOUSE(5))<=3
  0 AND act=0 THEN act=1:FR
  AMERECT VARPTR(asr(1))
WEND
lilw=(y2-33)*lilm/(hfen-64):
  IF act THEN si=1 ELSE asr
  (1)=yl:asr(3)=yl+16
WHILE si:si=0:w=yl<>y2 AND
  lilw<>lil OR yl=33 AND li
  l>0 OR yl=hfen-31 AND lil
  <lilm:IF w THEN lil=lilw:
  PENNORMAL:GOSUB aff ELS
  E FRAMERECT VARPTR(asr(
  1))
WEND:PENNORMAL:RETURN
Zon4:
mou=MOUSE(4):act=1
WHILE act:IF lil<lilm THEN
   lil=FNmin(lil+lipg-pas,li
  lm):GOSUB aff
act=MOUSE(0)<0 AND MOUSE(4
  )=mou AND MOUSE(6)>=asr(3
  )
WEND:RETURN
Zon5:
PUT(xfl,yfli),flin(1):act=1
WHILE act:IF lil<lilm THEN
   lil=lil+1:GOSUB aff
act=MOUSE(0)<0 AND MOUSE(6
  )>=yfli
WEND
PUT(xfl,yfli),flin(1):RETUR
  N
aff:
asr(1)=(hfen-64)*lil/lilm+33
  :asr(3)=asr(1)+16:FILLREC
  T VARPTR(bde(1)),VARPTR(
  gris(1)):ERASERECT VARPT
  R(asr(1)):FRAMERECT VARP
  TR(asr(1)):ON zon GOSUB A
  ff1,Aff2,Aff2,Aff2,Aff5:RE
  TURN
Aff1:
SCROLL(xlrol,ylrol)-(x2rol,
  y2rol),0,hli:MOVETO xli,y
  li:Afficher lil,1:RETURN
Aff2:
LINE(xlrol,ylrol)-(x2rol,y2r
  ol),30,bf:MOVETO xli,yli:
  Afficher lil,lipg:RETURN
Aff5:
SCROLL(xlrol,ylrol)-(x2rol,
  y2rol),0,-hli:MOVETO xli,
  yli+hli*(lipg-1): Afficher
   lil+lipg-1,1:RETURN
DATA 16,16,-1,&h8001,&h8101,
  &h8281,&h8441,&h8821,&h901
  1,&hA009,&hF83D,&h8821,&h8
  821,&h8821,&h8FE1,&h8001,&
  h8001,-1,16,16,0,0,0,&h100
  ,&h380,&h7C0,&hFE0,&h1FF0,
  &h7C0,&h7C0,&h7C0,&h7C0
END SUB
```

Compatible incompatible ? Suite...

(première partie dans le numéro 24)

Particularismes

Nous ne pouvons, faute de place, citer les particularismes de tous les logiciels sur le MacPlus, en dehors de leur appartenance aux cinq catégories décrites dans le numéro 24. C'est pourquoi nous citons seulement dans la liste ci-dessous les particularismes des logiciels les plus connus. Pour les logiciels listés dans Pom's 24, quand un numéro de version est indiqué, cela signifie en général que les versions antérieures posent encore plus de problèmes, ou ne marchent pas du tout, avec le MacPlus.

Basic 2.0 : selon Apple, il appartient à la catégorie 1 ; selon nous, à la catégorie 3, le Basic et les fichiers devant être au niveau 0 de la hiérarchie.

Jazz : ne pas mettre à jour la disquette originale ; constituer une nouvelle disquette de démarrage en 800Ko avec le système MacPlus ; y transférer les fichiers Convert, Jazz Ressources, Jazz System Update de la disquette de démarrage originale : le fichier Jazz Ressources doit être dans le dossier système.

MacDraft : reconnaît en partie le HFS, mais ne sait pas sauvegarder un document dans un dossier (le met au niveau 0). On peut utiliser une disquette 800Ko à condition de la formater en MFS.

MacTerminal 1.0 : mise à jour de la disquette possible, à condition de garder l'ancien clavier. La version 2.0 devrait être disponible en mai 86.

MacTracks : les fichiers auxiliaires doivent être dans le dossier système.

Microsoft File : problèmes de dossiers avec les index. Il est donc préférable de travailler au niveau 0 de la hiérarchie.

Multiplan : la version 1.10, sortie récemment et échangeable pour 250 F auprès de Microsoft, est compatible tous Macs, résout le problème de la version 1.02 avec le MacPlus (limite à 7K !) et ajoute les fonctions financières.

Omnis3 : légers problèmes avec le HFS ; les fichier auxiliaires doivent être dans le dossier système (Omnis3 Info).

Orthogiciel : nouvelle version attendue en juin 86.

Overvue 2.0 : selon Apple, appartient à la catégorie 1. En fait, ne tourne qu'avec son ancien système.

PageMaker 1.1 : légers problèmes avec le HFS. Les fichiers doivent être dans le même dossier que l'application. En cas d'utilisation avec la LaserWriter, l'imprimante doit absolument être baptisée "LaserWriter".

SmoothTalker 2.0 : aucun problème selon Apple (catégorie 1). Selon nous, plante complètement la machine.

Switcher 4.4 : les applications d'une liasse doivent être à la racine. Refuse le Finder. Conflits avec le cache-mémoire, qui doit donc être désactivé. Version 4.9 entièrement compatible attendue.

ThinkTank : problèmes avec le cache-mémoire. L'impression peut poser des problèmes aléatoires non résolus.

Word 1.02 : ne reconnaît pas le pavé numérique, mais accepte les chiffres du clavier principal. Nouvelle version incessamment. Le glossaire doit être dans le dossier de l'application.

Hervé Thiriez

Jean Damiènes
Alain Guillard
Paule Jade
Philippe Meslin
Hervé Thiriez

Des essais rapides

Lors de la mise au point d'un programme en assembleur, le passage par le 'Finder', au retour, fait perdre du temps. Pour éviter cette étape, on peut insérer ce petit segment de code à la place du classique RTS de fin de programme :

```
        LEA       Param,A0
        LEA       Program,A1
        MOVE.L    A1,(A0)
        _Launch   ; $A9F2
Param   DC.L      0
        DC        0
Program DC.B      4,'Edit',0
```

Ceci provoque le lancement de l'éditeur.

Un caractère retrouvé

Si vous êtes passé du Mac au MacPlus, vous avez certainement perdu le caractère "É", que l'on obtenait avec la séquence de touche :

Option ' E

Sur le Mac Plus, il faut faire :

Option 1 E

Qui a retrouvé la puce "•" ?

N.D.L.R. : le 'Mega-truc' en la matière, c'est l'accessoire "ad litteram" publié dans ce numéro (dommage pour l'accessoire "clavier", qui a quasiment disparu des disquettes de tous ceux qui ont essayé "ad litteram").

Clavier, clavier Plus ?

Si vous écrivez une application qui doit pouvoir fonctionner indifféremment sur un Mac ou sur un Mac Plus, il peut être intéressant de savoir à quel type de clavier on a affaire. Pour cela, on peut lire le contenu de l'adresse $21E, qui est égal à 11 en cas de présence d'un clavier "Mac Plus". En Basic :

```
IF PEEK(&h21E)=11 THEN
   MacPlus ELSE Mac
```

En assembleur :

```
        CMPI.B   #11,$21E
        BEQ      MacPlus
Mac     ...      ...
        ...      ...
```

Le collage dans Excel

Nous avons déjà constaté plusieurs bugs dans Excel; aucun d'entre eux n'est grave, mais ils sont quelque peu énervants. Un des plus originaux touche le *Collage Spécial*, qui – très intelligemment – permet de copier seulement les valeurs, les formules ou les formats mais, moins intelligemment, ne fait pas la distinction entre les formats et les encadrements.

Le bug est le suivant. Quand on copie le vecteur :

25	25	25	25

pour effectuer un collage spécial des valeurs seules et avec soustraction vers le vecteur :

17	0		30

on obtient le résultat :

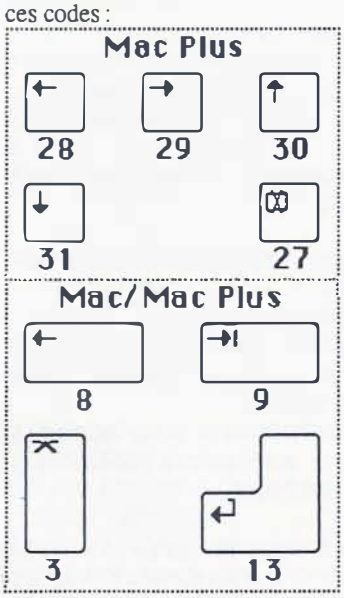
-8	-25	25	5

où la troisième valeur est erronée, car la cellule récipiendaire était vide.

Le seul remède que nous pouvons vous conseiller est de mettre des 0 au lieu de blancs dans les cellules appelées à recevoir des informations par collage spécial associé à une opération.

Touches spéciales Mac *Plus*

Le Mac *Plus* est arrivé avec quelques touches en *Plus* par rapport au Macintosh (sans parler des touches 'doublées' du pavé numérique). On notera que les touches communes aux deux appareils ont fort heureusement gardé les mêmes codes ASCII. Voici ces codes :



ROM 64Ko ou 128Ko ?

Le Macintosh Plus ne sera bientôt plus seul à disposer d'une ROM de 128Ko. En effet, le Macintosh 512/800 arrive... avec 128Ko de ROM. Les programmes qui aiment savoir où ils sont peuvent tester le contenu du mot de 16 bits situé à l'adresse $28E, qui est égal à $7FFF (32767) si l'on est en présence d'une ROM de 128Ko.

Exemple :

```
IF PEEK(&h28E)*256+
PEEK(&h28F)=32767 THEN
ROM128 ELSE ROM64
```

Avez-vous la disquette Mac 25 ?

Au sommaire : µFinder, Ad litteram, les ascenseurs en Basic, Système version 3.2c et Finder 5.3

Francophonisme... Suite.

*Suite à la question métaphysique posée dans le numéro 24 – rappelons qu'il s'agissait de trouver un exemple de texte français comportant tout l'alphabet en une phrase aussi courte que possible –, Georges Boyer, un lecteur d'Annecy nous propose : **"Portez ce bol de vieux whisky aux juges qui fument"***

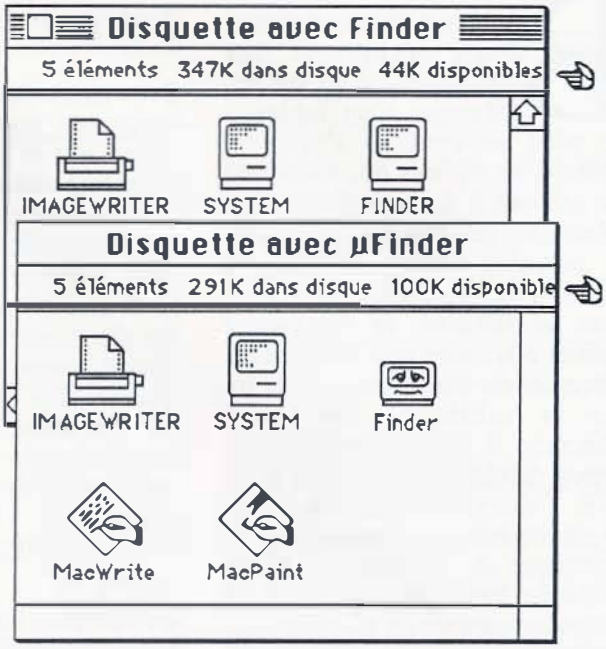
*Il nous suggère de le traduire pour les américains par "Bring this cup of old whisky..." ! Les habitués des connexions Transpac connaissent certainement le : **"Voyez le brick géant que j'examine près du wharf".***

Gagner 56 Ko ? : µFINDER

Jean-Luc Bazanegue

Le Finder du Macintosh est une application sophistiquée qui autorise une gestion complète des fichiers placés sur nos disquettes. A l'usage, on s'aperçoit que la plupart des fonctions sont rarement utilisées lorsqu'on emploie, par exemple, des utilitaires (PaintMover, REdit...). De plus, le passage par le Finder prend du temps et, dans le cas de programme réclamant beaucoup de place sur la disquette (comme MacPaint), il peut être intéressant de récupérer la place qu'il occupe.

Nous vous proposons donc un "micro-Finder", qui ne conserve que les fonctions essentielles de son grand frère (sélection d'une application et éjection/redémarrage). On utilise pour cela la fenêtre de sélection du système (dont l'aspect peut varier d'un système à l'autre), ce qui nous permet de gagner 56Ko sur chaque disquette et, du même coup, beaucoup de temps lors du passage d'une application à l'autre.



Fichier 'µFinder.Job'

```
Asm     µFinder.Asm   Exec    Edit
Link    µFinder.Link  Exec    Edit
RMaker  µFinder.R     Finder  Edit
```

Fichier 'µFinder.Link'

```
[
µFinder
/Output µFinder.Code
/Type 'TEMP'
$
```

Fichier 'µFinder.R'

```
µFinder
APPLµFND
Type µFND = STR
   ,0
µFinder - Pom's- Juin 86
Type BNDL
   ,128(32)
   µFND 0
   ICN#
   0 128
   FREF
   0 128
Type FREF
   ,128(32)
   APPL 0
Type ICN# = GNRL
   ,128(32)
2
00000000   00000000
00000000   00000000
00000000   00000000
00000000   00000000
00000000   00000000
00000000   00000000
00000000   07FFFFE0
08000010   08FFFF10
09000090   09024090
091E7890   09366C90
095E7A90   090C3090
09000090   08FFFF10
08000010   08100810
082FF410   08000010
08000010   07FFFFE0
04000020   07FFFFE0
*
00000000   00000000
00000000   00000000
00000000   00000000
00000000   00000000
00000000   00000000
00000000   00000000
00000000   07FFFFE0
0FFFFFF0   0FFFFFF0
0FFFFFF0   0FFFFFF0
0FFFFFF0   0FFFFFF0
0FFFFFF0   0FFFFFF0
0FFFFFF0   0FFFFFF0
0FFFFFF0   0FFFFFF0
0FFFFFF0   07FFFFE0
07FFFFE0   07FFFFE0
INCLUDE µFinder.Code
```

Source 'µFinder.Asm'

```
; Routines utilisées:
.Trap   _Eject          $A017
.Trap   _FlushEvents    $A032
.Trap   _GetCursor      $A9B9
.Trap   _InitCursor     $A850
.Trap   _InitDialogs    $A97B
.Trap   _InitFonts      $A8FE
.Trap   _InitGraf       $A86E
.Trap   _InitMenus      $A930
.Trap   _InitWindows    $A912
.Trap   _Launch         $A9F2
.Trap   _Pack3          $A9EA
.Trap   _SetCursor      $A851
.Trap   _SetHandleSize  $A024
.Trap   _SetVol         $A015
.Trap   _TEInit         $A9CC
; Equivalences :
rGood          EQU   0
rVolume        EQU   6
rName          EQU   10
SFGetFile      EQU   2
ioNamePtr      EQU   $12
ioVDrvNum      EQU   $16
AppParmHandle  EQU   $AEC
watchCursor    EQU   4
ioVQElSize     EQU   $40
; Initialisations :
        PEA     -4(A5)
        _InitGraf
        _InitFonts
        MOVE.L  #$0000FFFF,D0
        _FlushEvents
        _InitWindows
        _InitMenus
        CLR.L   -(SP)
        _InitDialogs
        _TEInit
        _InitCursor
; Affichage de la fenêtre de sélection
; standard
        MOVE    #80,-(SP)
        MOVE    (SP),-(SP)
        CLR.L   -(SP)
        CLR.L   -(SP)
        MOVE    #1,-(SP)
        PEA     type
        CLR.L   -(SP)
        PEA     reponse(A5)
        MOVE    #SFGetFile,-(SP)
        _Pack3
; Curseur en forme de montre
        MOVE    #watchCursor,-(SP)
        _GetCursor
        MOVEA.L (SP)+,A2
        MOVE.L  (A2),-(SP)
        _SetCursor
;Prépare le tampon
; d'entrées/sorties
        LEA     IO,A0
        CLR.L   ioNamePtr(A0)
        TST     Reponse+rGood(A5)
        BNE.S   @1
; Si on demande "Annuler", les
; disquettes sont éjectées et on
; redémarre le système
        MOVE    #-1,ioVDrvNum(A0)
        _Eject
        MOVE    #-2,ioVDrvNum(A0)
        _Eject
        JMP     $40000A
; Indique le volume où se trouve
; l'application sélectionnée comme
; étant le volume par défaut
@1      MOVE    Reponse+rVolume(A5),
                ioVDrvNum(A0)
        _SetVol
; Pour indiquer à l'application
; sélectionnée qu'il n'y a rien à ouvrir ou
; imprimer (voir pom's24)
        MOVEQ.L #4,D0
        MOVE.L  AppParmHandle,A0
        _SetHandleSize
        MOVE.L  AppParmHandle,A0
        MOVE.L  (A0),A0
        CLR.L   (A0)
; Lance l'application demandée
        MOVE    #0,-(SP)
        PEA     reponse+rName(A5)
        MOVE.L  SP,A0
        _Launch
;Tampons et constantes
type    DC.B    'APPL'
IO      DC.B    ioVQElSize,0
Reponse DS.B    42
```

Comment faire ?

vous avez la disquette Mac25:

- démarrez le système avec la disquette Mac 25 ;
- insérez la disquette sur laquelle vous désirez placer le µFinder ;
- supprimez le vrai Finder ;
- faites une copie du µFinder sur la disquette destination ;
- supprimez le 'µ' de µFinder.

vous n'avez pas la disquette Mac25:

- munissez-vous de votre système de développement 68000 ;
- saisissez les fichiers et le source listés ci-contre ;
- lancer l'EXECution du fichier 'µFinderJob' ;
- faites alors comme si vous aviez la disquette 25.

Tri, Chaînes, Index

Bernard Lambillon

Beaucoup de programmes Basic nécessitent des opérations de tri sur les éléments d'un tableau. Le plus souvent, il s'agit de tableaux de chaînes de caractères. Un tableau à deux indices peut mémoriser les adresses de clients, le premier indice repère la position de l'adresse du client dans le tableau, le deuxième permet d'accéder aux différentes rubriques de son adresse. Si le tri sur la rubrique 'nom' est effectué, il va de soi que les autres rubriques associées à un nom doivent suivre les déplacements imposés par la technique de tri choisie. Cette méthode implique une perte de temps importante du fait que tous les éléments du tableau participent à l'opération de tri. Pour éviter ce grand "chambardement", on décide de ne plus modifier le tableau. L'ordre des éléments du tableau de chaînes est enregistré dans une tableau de nombres, appelé table d'index. L'accès à la liste triée se fait alors via cette table d'index. Considérons un tri sur le nom des pays du tableau PAY$.

Tableau PAY$

				P% après tri
I = 0				0
I = 1	Italie	Rome	I	7
I = 2	Pays-Bas	La Haye	NL	3
I = 3	Belgique	Bruxelles	B	5
I = 4	France	Paris	F	4
I = 5	Danemark	Copenhague	DK	8
I = 6	Luxembourg	Luxembourg	L	9
I = 7	Allemagne	Bonn	D	10
I = 8	Grande-Bretagne	Londres	GB	1
I = 9	Grèce	Athènes	G	6
I = 10	Irlande	Dublin	IRL	2

Il suffit alors de lire la table d'index P% pour obtenir la liste des pays classés par ordre alphabétique :

```
FOR I=1 TO N
J = P%(I)
PRINT PAY$(J,0); TAB( 20);
PAY$(J,1); TAB( 34);PAY$(J,2)
NEXT I
```

Tri par insertion

Cette technique de tri peut se comparer avec l'insertion des cartes lors d'une donne. Dès la deuxième carte, le joueur insère la carte donnée en la comparant à la (aux) précédente(s). Tant que la carte donnée est inférieure à la carte comparée, les cartes sont décalées vers la droite et la comparaison se poursuit vers la gauche. Dès que la carte donnée est supérieure ou égale à la carte comparée, la carte donnée est insérée à droite de la carte comparée. Le programme TRI.CHAINE.DEMO illustre ce type de tri.

Le tri par insertion est assez performant (au niveau de la rapidité) compte tenu du peu d'instructions qu'il requiert. Notons cependant que l'élément d'indice 0 doit contenir la plus petite valeur pour le type de variable triée. Ainsi, pour des chaînes de caractères, la chaîne vide est cette valeur qui est placée automatiquement lors du dimensionnement du tableau. Pour un tableau de nombres positifs, 0 est une valeur inférieure ou égale à tout nombre positif. Par contre, pour un tableau de nombres positifs, nuls ou négatifs, il est obligatoire d'assigner respectivement -32767 ou -10E38 à l'élément 0 du tableau d'entiers ou du tableau de réels. L'élément 0 du tableau ne peut pas être utilisé pour inscrire des informations à trier.

Le tri du tableau PAY$ peut s'écrire en Basic (voir le programme TRI CHAINE BASIC.1.1) comme ceci : →

Tri en langage machine

Au niveau de la structure, le programme TRI.CHAINE.C est l'équivalent du programme précédent mais il est écrit en codes 6502. Il occupe 215 octets de mémoire. Il est bien sûr plus rapide : une liste de 500 noms de 24 caractères est triée en 10 secondes (cf. TRI.CHAINE.1.2) au lieu de 35 minutes en Basic !

La syntaxe est la suivante : &N,PAY$(1,CLE),P%(1) où N contient le nombre d'éléments à trier et CLE vaut 0,1 ou 2 selon que le tri s'effectue sur le nom, la capitale ou l'abréviation du pays. Voici d'autres exemples :

& 500,a$(1),IX%(1)
& t(0),ad$(1,3),h%(1,3)

Le chargement du programme s'effectue automatiquement à l'aide des programmes TRI.DOS.INIT.C ou TRI.PRO.INIT.C suivant que le système est géré sous DOS 3.3 ou ProDOS. Dans la version DOS, le chargement doit s'effectuer avant toute création de chaînes de caractères car le programme se loge sous le HIMEM. Par contre, en ProDOS, le programme TRI.CHAINE.C est chargé entre le Basic.System et sa mémoire tampon d'usage général sans altérer les zones réservées au Basic.

Le programme source, écrit avec l'assembleur ProCODE, est documenté de telle sorte que le lecteur puisse faire le parallèle entre le programme de tri en

```
900 P%(1)=1                                   Init table index
1000 FOR I = 2 TO N                           Boucle externe
1010 X$ = PAY$(I,CLE) :                       Extraction
     P%(I) = I
1020 FOR J = 1 TO 0 STEP -1                   Boucle interne
1030 JJ = P%(J)                               Entier él. comparé
1040 ON 2 - (X$ > PAY$                        Comparaison
     (JJ,CLE)) GOSUB 200
     0,3000
                                              Si... ALORS insertion SINON décalage
1050 NEXT J
1060 NEXT I

2000 REM Insertion
2020 P%(J+1) = I
2030 J = 0
2040 RETURN

3000 REM décalages
3020 P%(J+1) = JJ
3030 RETURN
```

Basic et celui-ci. Il suffit de se rappeler que chaque élément d'un tableau de chaînes occupe 3 octets et que chaque nombre d'un tableau d'entiers en nécessite 2.

longueur de la chaîne	adresse basse de stockage	adresse haute de stockage	partie haute de l'entier	partie basse de l'entier

Voici la situation avant le tri (programme TRI.CHAINE.1.0) :

Page zéro

```
 6: CE 14   Pointeur él 0 de PAY$ (14CE)
 8: D1 14   Pointeur él 1 de PAY$ (14D1)
83: 3A 15   Pointeur él 1 de P% (153A)
50: 0A 00   Nb d'éléments à trier (000A)
```

Tableau PAY$

```
                                        Longueur-Pointeur
14CE: 00 00 00 él0 Vide
14D1: 06 01 12 él1 Italie          6        $1201
14D4: 0B 1B 12 él2 Pays-Bas        B        $121B
14EC: 07 28 13 él3 Belgique        8        $123B
14DA:
14EC: 07 28 13  él10. Irlande      7        $1328
```

Tableau P%

```
153B: 00 00    élément  0: Valeur 0
153A: 00 01    élément  1: Valeur 1
153D: 00 00    élément  2: Valeur 0
1540:
154C: 00 00    élément 10: Valeur 0
```

Zone chaines de caractères (dans les DATAs du programme)

```
1201: 49 74 61 6C 69 65 22 2C 22 ...
      I  T  a  l  i  e  "  ,  "  _
121B: 50 61 79 73 2D 42 61 73 22 2C 22...
      P  a  y  s  -  B  a  s  "  ,  "  _
123B: 42 65 6C 67 69 71 75 65...
      B  e  l  g  i  q  u  e  ...
125C: 46 72 61 6E 63 65...
      F  r  a  n  c  e
1328: 49 72 6C 61 6E 64 65...
      I  r  l  a  n  d  e
```

Programme 'TRI.CHAINE.MENU'

```
10  REM  *** TRI.CHAINE.MENU ***
20 :
30  GOSUB 70: REM INITIALISATION
40  IF R > 0 THEN  GOSUB 720
50  HOME : END
60 :
70  REM  /// INITIALISATION ///
80 :
90  PRINT  CHR$ (21)
100 :
110  REM  -- LECTURE DATA --
120 :
130  DIM M$(5)
140  FOR I = 0 TO 5
150  READ M$(I)
160  NEXT I
170  DIM L$(8)
180  FOR I = 1 TO 8
190  READ L$(I)
200  NEXT I
210 :
220  REM  -- DOS/PRODOS --
230 :
240  DIM RU$(1)
250  READ RU$(0),RU$(1)
260 PRO =  PEEK (48896) = 76
270 :
280  REM  -- CONSTANTES --
290 :
300 D$ =  CHR$ (4)
310 BR$ =  CHR$ (7)
320 L$ = "____________________________________
____________"
330 L1 = 20 -  LEN (M$(0)) / 2
340 L2 = L1 + ( INT (L1) / 2 <  > L1 /
    2)
350 :
360  REM  -- TITRE --
370 :
380  TEXT : HOME
390  INVERSE
400  PRINT  SPC( L1);M$(0); SPC( L2)
410 :
420  REM  -- BAS ECRAN --
430 :
440  VTAB 24: PRINT  SPC( 39)
450  POKE 2039,32
460  VTAB 24: HTAB 5: PRINT M$(1);
470 H = 5 +  LEN (M$(1))
480  NORMAL
490 :
500  REM  -- MENU --
510 :
520  VTAB 2: HTAB 1: PRINT L$;
530  FOR I = 1 TO 4
540  PRINT
550  HTAB 8
560  PRINT "<";I;"> ";M$(1 + I)
570  HTAB 12: PRINT L$(2 * I - 1)
580  HTAB 12: PRINT L$(2 * I)
590  PRINT L$;
600  NEXT I
610 :
620  REM  -- CHOIX --
630 :
640  VTAB 24: HTAB H
650  GET R$:R =  VAL (R$)
660  IF R >  = 1 AND R <  = 4 OR R$ =
     CHR$ (27) THEN 700
670  PRINT BR$;
680  GET R$:R =  VAL (R$)
690  GOTO 660
700  RETURN : REM <INITIALISATION>
710 :
720  REM  /// PROGRAMME ///
730 :
740  VTAB 5 * R - 1: HTAB 12
750  INVERSE
760  PRINT M$(1 + R)
770  NORMAL
780  PRINT D$;RU$(PRO);M$(1 + R)
790  RETURN : REM  <PROGRAMME>
800 :
810  REM  ### DONNEES FIXES ###
820 :
830  DATA "TRI DE TABLEAU DE CHAINES"
840  DATA  "VOTRE CHOIX (1-4 OU ESC):
     "
850  DATA "TRI.CHAINE.1.0"
860  DATA "TRI.CHAINE.1.1"
870  DATA "TRI.CHAINE.1.2"
880  DATA "TRI.CHAINE.DEMO"
890  DATA "tri d'un tableau 10X3"
900  DATA "en langage-machine"
910  DATA "tri d'un tableau 10X3"
920  DATA "en Basic."
930  DATA "tri de 500 noms de 24 car."
940  DATA "en langage-machine (10 sec.
     )"
950  DATA "Animation de la technique"
960  DATA "de tri par insertion"
970  DATA "RUN"
980  DATA "RUN BASIC/"
```

Programme 'TRI.CHAINE.1.0'

```
10  REM  *** TRI.CHAINE.1.0 ***
20 :
30  GOSUB 150: REM  /// INIT PROGRAMME
    ///
40 :
50  REM  ::: REPETER :::
60 :
70  GOSUB 670: REM  /// MENU ///
80  IF  NOT FINI THEN 50
90 :
100  REM  ::: FIN REPETER :::
110 :
120  IF CAS = 4 THEN 1500: REM  <RETOU
     R MENU>
130  END
140 :
150  REM  /// INIT PROGRAMME ///
160 :
170  REM  -- DOS OU PRODOS --
180 :
190 DOS =  PEEK (48896) <  > 76
200  ON 2 - DOS GOSUB 530,590
210 :
220  REM  -- CAR. DE CONTROLE --
230 :
240 B$ =  CHR$ (8)
250 BR$ =  CHR$ (7)
260 :
270  REM  -- TABLES DES PAYS --
280 :
290  DIM PAY$(10,2),P%(10)
300 :
310  READ N
320  FOR I = 1 TO N
330  FOR J = 0 TO 2
340  READ PAY$(I,J)
350  NEXT J
360  NEXT I
370 :
380  REM  -- TABLE DES MESSAGES --
390 :
400  DIM M$(5)
410  FOR M = 0 TO 5
420  READ M$(M)
430  NEXT M
440 :
450  REM  -- AFFICHAGE LISTE A TRIER
460 :
470  PRINT  CHR$ (21): REM  <CARTE 80
     COL. DESACTIVEE>
480  TEXT : HOME
```

```
490 V = 1: GOSUB 1330: REM   /// AFFIC
    HAGE LISTE ///
500  POKE 34,12
510  RETURN : REM  <INIT PROGRAMME>
520 :
530  REM  /// INIT DOS ///
540 :
550  PRINT  CHR$ (4);"BRUN TRI.DOS.INI
     T.C"
560 D$ =  CHR$ (13) +  CHR$ (4)
570  RETURN : REM  <INIT DOS>
580 :
590  REM  /// INIT PRODOS ///
600 :
610 D$ =  CHR$ (4)
620  PRINT D$;"PREFIX CODE.MACHINE"
630  PRINT D$;"BRUN TRI.PRD.INIT.C"
640  PRINT D$;"PREFIX /"
650  RETURN : REM  <INIT PRODOS>
660 :
670  REM  /// MENU ///
680 :
690  REM  -- AFFICHAGE MENU --
700 :
710  HOME
720  FOR M = 0 TO 5
730  VTAB 13 + 2 * M: HTAB 5: PRINT M$
     (M);
740  NEXT M
750 :
760  REM  -- INIT CHOIX --
770 :
780  POKE 49168,0
790  PRINT B$;B$;
800 :
810  REM  -- CHOIX --
820 :
830  GET R$
840  IF  ASC (R$) < 32 THEN R$ = "?"
850  PRINT R$;B$;
860 C1 = (R$ = "P" OR R$ = "p")
870 C2 = (R$ = "C" OR R$ = "c")
880 C3 = (R$ = "A" OR R$ = "a")
890 C4 = (R$ = "M" OR R$ = "m")
900 C5 = (R$ = "S" OR R$ = "s")
910 CAS = C1 + C2 * 2 + C3 * 3 + C4 *
    4 + C5 * 5
920  IF CAS = 0 THEN 810
930 :
940  REM  -- TRAITEMENTS --
950 :
960  ON CAS GOSUB 990,990,990,1150,115
     0
970  RETURN : REM  <AFFICHAGE MENU>
980 :
990  REM  /// TRI ///
1000 :
1010 CLE = CAS - 1
1020  & N,PAY$(1,CLE),P%(1)
1030 TRIE = 1
1040 :
1050  REM  -- AFFICHAGE LISTE TRIEE --
1060 :
1070  HOME
1080 V = 13: GOSUB 1330: REM  /// AFFI
     CHAGE LISTE ///
1090 :
1100  REM  -- ATTENTE TOUCHE --
1110 :
1120  GOSUB 1420: REM /// TOUCHE ///
1130  RETURN : REM  <TRI>
1140 :
1150  REM  /// FINI ///
1160 :
1170 FINI = 1
1180  POKE 34,0
1190  VTAB 23
1200  ON 2 - DOS GOSUB 1230,1280
1210  RETURN : REM  <FINI>
1220 :
1230  REM  /// FIN DOS ///
1240 :
1250  HIMEM: 38400
1260  RETURN : REM <FIN DOS>
1270 :
1280  REM  /// FIN PRODOS ///
1290 :
1300  CALL 48888: REM  <LIBERATION DES
      ZONES ALLOUEES ($BEF8: FREEBFR)
1310  RETURN : REM  <FIN PRODOS>
1320 :
1330  REM  /// AFFICHAGE LISTE ///
1340 :
1350  VTAB V: HTAB 1
1360  FOR I = 1 TO N
1370 J = P%(I) * TRIE + I *  NOT TRIE
1380  PRINT PAY$(J,0); TAB( 20);PAY$(J
      ,1); TAB( 34);PAY$(J,2)
1390  NEXT I
1400  RETURN : REM  AFFICHAGE LISTE
1410 :
1420  REM  /// TOUCHE ///
1430 :
1440  PRINT  CHR$ (7);
1450  POKE 49168,0: WAIT 49152,128: PO
      KE 49168,0
1460  PRINT  CHR$ (7);
1470 :
1480  RETURN : REM  <TOUCHE>
1490 :
1500  REM  /// RETOUR MENU ///
1510 :
1520  READ M$
1530  POKE 34,0
1540  VTAB 1: PRINT
1550  ON 2 - DOS GOTO 1560,1570
1560  PRINT D$;"RUN";M$
1570  PRINT D$;"RUN BASIC/";M$
1580 :
1590  REM  ### DONNEES FIXES ###
1600 :
1610  DATA  10
1620 :
1630  REM  -- NOM DES PAYS --
1640 :
1650  DATA "Italie","Rome","I"
1660  DATA "Pays-Bas","La Haye","NL"
1670  DATA "Belgique","Bruxelles","B"
1680  DATA "France","Paris","F"
1690  DATA "Danemark","Copenhague","DK
      "
1700  DATA "Luxembourg","Luxembourg","
      L"
1710  DATA "Allemagne fédérale","Bonn"
      ,"D"
1720  DATA "Grande Bretagne","Londres"
      ,"GB"
1730  DATA "Grèce","Athènes","G"
1740  DATA "Irlande","Dublin","IRL"
1750 :
1760  REM  -- MESSAGES --
1770 :
1780  DATA "Tri par ..... <P>ays."
1790  DATA "Tri par ..... <C>apitale."
1800  DATA "Tri par ..... <A>bréviatio
      n."
1810  DATA "Retour au ... <M>enu."
1820  DATA "..............<S>ortie."
1830  DATA "Votre choix:  <?>"
1840 :
1850  REM  -- MENU --
1860 :
1870  DATA "TRI.CHAINE.MENU"
```

Programme 'TRI.CHAINE.1.1'

```
10  REM *** TRI.CHAINE.1.1 ***
20 :
30  GOSUB 150: REM  /// INIT PROGRAMME
     ///
40 :
50  REM  ::: REPETER :::
60 :
70  GOSUB 490: REM /// MENU ///
80  IF  NOT FINI THEN 50
90 :
100  REM  ::: FIN REPETER :::
110 :
120  IF CAS = 4 THEN 1580: REM  <RETOU
     R MENU>
130  END
140 :
150  REM  /// INIT PROGRAMME ///
160 :
170  REM  -- CAR. DE CONTROLE --
180 :
190 D$ =  CHR$ (4)
200 B$ =  CHR$ (8)
210 BR$ =  CHR$ (7)
220 :
230  REM  -- TABLES DES PAYS --
240 :
250  DIM PAY$(10,2),P%(10)
260 :
270  READ N
280  FOR I = 1 TO N
290  FOR J = 0 TO 2
300  READ PAY$(I,J)
310  NEXT J
320  NEXT I
330 :
340  REM  -- TABLE DES MESSAGES --
350 :
360  DIM M$(5)
370  FOR M = 0 TO 5
380  READ M$(M)
390  NEXT M
400 :
410  REM  -- AFFICHAGE LISTE A TRIER
420 :
430  PRINT  CHR$ (21): REM  <CARTE 80
     COL. DESACTIVEE>
440  TEXT : HOME
450 V = 1: GOSUB 1410: REM  /// AFFIC
     HAGE LISTE ///
460  POKE 34,12
470  RETURN : REM  <INIT PROGRAMME>
480 :
490  REM  /// MENU ///
500 :
510  REM  -- AFFICHAGE MENU --
520 :
530  HOME
540  FOR M = 0 TO 5
550  VTAB 13 + 2 * M: HTAB 5: PRINT M$
     (M);
560  NEXT M
570 :
580  REM  -- INIT CHOIX --
590 :
600  POKE 49168,0
610  PRINT B$;B$;
620 :
630  REM  -- CHOIX --
640 :
650  GET R$
660  IF  ASC (R$) < 32 THEN R$ = "?"
670  PRINT R$;B$;
680 C1 = (R$ = "P" OR R$ = "p")
690 C2 = (R$ = "C" OR R$ = "c")
700 C3 = (R$ = "A" OR R$ = "a")
```

```
710 C4 = (R$ = "M" OR R$ = "m")
720 C5 = (R$ = "S" OR R$ = "s")
730 CAS = C1 + C2 * 2 + C3 * 3 + C4 *
    4 + C5 * 5
740  IF CAS = 0 THEN 630
750 :
760  REM  -- TRAITEMENTS --
770 :
780  ON CAS GOSUB 810,810,810,1340,134
    0
790  RETURN : REM  <AFFICHAGE MENU>
800 :
810  REM  /// TRI ///
820 :
830  REM  -- INIT TRI --
840 :
850 CLE = CAS - 1
860 P%(1) = 1
870 :
880  REM  -- BOUCLE EXTERNE --
890 :
900  FOR I = 2 TO N
910 :
920  REM  -- EXTRACTION --
930 :
940 X$ = PAY$(I,CLE)
950 :
960  REM  -- BOUCLE INTERNE --
970 :
980  FOR J = I - 1 TO 0 STEP  - 1
990 JJ = P%(J)
1000 :
1010  REM  -- COMPARAISON --
1020 :
1030  ON 2 - (PAY$(JJ,CLE) < X$) GOSUB
     1120,1180
1040 :
1050  NEXT J,I
1060 :
1070  REM  -- AFFICHAGE --
1080 :
1090  GOSUB 1230: REM /// AFFICHAGE LI
     STE TRIEE ///
1100  RETURN : REM <TRI>
1110 :
1120  REM  /// INSERTION ///
1130 :
1140 P%(J + 1) = I
1150 J = 0
1160  RETURN : REM  <INSERTIOH>
1170 :
1180  REM  /// DECALAGE ///
1190 :
1200 P%(J + I) = JJ
1210  RETURN : REM  <DECALAGE>
1220 :
1230  REM / AFFICHAGE LISTE TRIEE /
1240 :
1250  HOME
1260 TRIE = 1
1270 V = 13: GOSUB 1410: REM  /// AFFI
     CHAGE LISTE ///
1280 :
1290  REM  -- ATTENTE TOUCHE --
1300 :
1310  GOSUB 1500: REM /// TOUCHE ///
1320  RETURN : REM  <TRI>
1330 :
1340  REM  /// FINI ///
1350 :
1360 FINI = 1
1370  POKE 34,0
1380  VTAB 23
1390  RETURN : REM  <FINI>
1400 :
1410  REM  /// AFFICHAGE LISTE ///
1420 :
1430  VTAB V: HTAB 1
1440  FOR I = 1 TO N
1450 J = P%(I) * TRIE + I *  NOT TRIE
1460  PRINT PAY$(J,0); TAB( 20);PAY$(J
     ,1); TAB( 34);PAY$(J,2)
1470  NEXT I
1480  RETURN : REM  <AFFICHAGE LISTE>
1490 :
1500  REM  /// TOUCHE ///
1510 :
1520  PRINT  CHR$ (7);
1530  POKE 49168,0: WAIT 49152,128: PO
     KE 49168,0
1540  PRINT  CHR$ (7);
1550 :
1560  RETURN : REM  <TOUCHE>
1570 :
1580  REM  /// RETOUR MENU ///
1590 :
1600  READ M$
1610  POKE 34,0
1620  VTAB 1: PRINT
1630 PRO =  PEEK (48896) = 76
1640  ON 2 - PRO GOTO 1650,1660
1650  PRINT D$;"RUN BASIC/";M$
1660  PRINT D$;"RUN";M$
1670 :
1680  REM  ### DONNEES FIXES ###
1690 :
1700  DATA  10
1710 :
1720  REM  -- NOM DES PAYS --
1730 :
1740  DATA "Italie","Rome","I"
1750  DATA "Pays-Bas","La Haye","NL"
1760 :
1770  DATA "Belgique","Bruxelles","B"
1770  DATA "France","Paris","F"
1790  DATA "Danemark","Copenhague","DK
     "
1800  DATA "Luxembourg","Luxembourg","
     L"
1810  DATA "Allemagne fédérale","Bonn"
     ,"D"
1820  DATA "Grande Bretagne","Londres"
     ,"GB"
1830  DATA "Grèce","Athènes","G"
1840  DATA "Irlande","Dublin","IRL"
1850 :
1860  REM  -- MESSAGES --
1870 :
1880  DATA "Tri par ..... <P>ays."
1890  DATA "Tri par ..... <C>apitale."
1900  DATA "Tri par ..... <A>bréviatio
     n."
1910  DATA "Retour au ... <M>enu."
1920  DATA ".............<S>ortie."
1930  DATA "Votre choix:  <?>"
1940 :
1950  REM  -- MENU --
1960 :
1970  DATA "TRI.CHAINE.MENU"
```

Programme 'TRI.CHAINE.1.2'

```
10  REM  *** TRI.CHAINE.1.2 ***
20 :
30  GOSUB 150: REM  /// INIT ///
40 :
50  FOR I = 1 TO N
60  GOSUB 580: REM  /// LECTURE ///
70  GOSUB 640: REM  // AFFICHAGE //
80  NEXT 1
90 :
100  GOSUB 710: REM  /// TRI ///
110  GOSUB 790: REM  LISTE TRIEE
120  GOSUB 920: REM  /// FIN ///
130  END
140 :
150  REM  /// INIT ///
160 :
170  PRINT  CHR$ (21)
180 :
190  REM  -- DOS OU PRODOS --
200 :
210 DOS =  PEEK (48896) <  > 76
220  ON 2 - DOS GOSUB 440,500
230 :
240  REM - TABLEAUX ET CONSTANTES -
250 :
260  DIM N$(500),IX%(500)
270  READ M$,NO$,F$,T$,S$
280 :
290  REM  -- MASQUE D'ECRAN --
300 :
310  SPEED= 255: TEXT : HOME
320  VTAB 12: HTAB 5: PRINT M$
330 L = 5 +  LEN (M$)
340  VTAB 14: HTAB 5: PRINT NO$
350 :
360  REM  -- FICHIER EN ENTREE --
370 :
380  IF  NOT DOS THEN F$ = "FICHIERS/"
     + F$
390  PRINT D$;"OPEN";F$
400  PRINT D$;"READ";F$
410  INPUT N
420  RETURN : REM  <INIT>
430 :
440  REM  /// INIT DOS ///
450 :
460  PRINT  CHR$ (4);"BRUN TRI.DOS.INI
     T.C"
470 D$ =  CHR$ (13) +  CHR$ (4)
480  RETURN : REM  <INIT DOS>
490 :
500  REM /// INIT PRODOS ///
510 :
520 D$ =  CHR$ (4)
530  PRINT D$;"PREFIX CODE.MACHINE"
540  PRINT D$;"BRUN TRI.PRO.INIT.C"
550  PRINT D$;"PREFIX /"
560  RETURN : REM  <INIT PRODOS>
570 :
580  REM  /// LECTURE ///
590 :
600  PRINT D$;"READ";F$
610  INPUT N$(I)
620  RETURN : REM  <LECTURE>
630 :
640  REM  /// AFFICHAGE ///
650 :
660  VTAB 12: HTAB L: CALL 64668
670  PRINT N$(I)
680  VTAB 14: HTAB L: PRINT I
690  RETURN : REM  <AFFICHAGE>
700 :
710  REM  /// TRI ///
720 :
730  HOME
740  HTAB 21 -  LEN (T$) / 2
750  VTAB 12: PRINT T$
760  & N,N$(1),IX%(1)
770  RETURN : REM  <TRI>
780 :
790  REM  /// LISTE TRIEE ///
800 :
810  SPEED= 220: HOME
820  PRINT " NO"; TAB( 9);"NOMS"; TAB(
     28);"NO DE FICHE"
830  PRINT "::::::::::::::::::::::::::
     ::::::::::::::";
840  POKE 34,2: VTAB 3
850  FOR I = 1 TO N
860  HTAB 4 -  LEN ( STR$ (I)): PRINT
     I;
870  HTAB 7: PRINT N$(IX%(I));
```

```
880  HTAB 34 -  LEN ( STR$ (IX%(I))):
     PRINT IX%(1)
890  NEXT 1
900  RETURN : REM  <LISTE TRIEE>
910 :
920  REM  /// FIN ///
930 :
940  PRINT D$;"CLOSE";F$
950  ON 2 - DOS GOSUB 1040,1090
960  SPEED= 255
970  PRINT : PRINT
980  HTAB 21 -  LEN (S$) / 2
990  VTAB 23: PRINT S$: POKE 34,0
1000  POKE 49168,0: WAIT 49152,128: PO
     KE 49168,0
1010  IF  PEEK (49152) <  > 27 THEN  G
     OSUB 1140: REM /// MENU ///
1020  RETURN : REM  <FIN>
1030 :
1040  REM  /// FIN DOS ///
1050 :
1060  HIMEM: 38400
1070  RETURN : REM <FIN DOS>
1080 :
1090  REM  /// FIN PRODOS ///
1100 :
1110  CALL 48888: REM  <DESALLOUE LES
     ZONES DE MEMOIRE-TAMPON>
1120  RETURN : REM <FIN PRODOS>
1130 :
1140  REM  /// MENU ///
1150 :
1160  ON 2 - DOS GOTO 1170,1180
1170  PRINT D$;"RUN TRI.CHAINE.MENU"
1180  PRINT D$;"RUN BASIC/TRI.CHAINE.M
     ENU"
1190 :
1200  REM  ### DONNEES FIXES ###
1210 :
1220  DATA "LECTURE DU NOM : "
1230  DATA "FICHE NO        : "
1240  DATA "TRI.LISTE"
1250  DATA "TRI EN COURS"
1260  DATA "ESC: SORTIE / AUTRE TOUCHE
     : MENU"
```

Programme 'TRI.CHAINE.DEMO'

```
10  REM *** TRI.CHAINE.DEMO ***
20 :
30  REM  ### CORPS DU PROGRAMME ###
40 :
50  GOSUB 2200: REM  /// INITIALISATIO
     N ///
60  GOSUB 1770: REM /// TRI PAR INSERT
     ION ///
70  GOSUB 1870: REM  /// FIN DE PROGRA
     MME ///
80  END
90 :
100  REM / AFFICHAGE DES MESSAGES /
110 :
120  VTAB 23: CALL  - 868: NORMAL
130  HTAB 19 -  LEN (M$(M)) / 2: PRINT
      M$(M)
140  RETURN
150 :
160  REM  /// PAUSE ///
170 :
180  REM   MODIF. DUREE DES PAUSE
190 :
200  REM  --- LECTURE CLAVIER ---
210 :
220 X =  PEEK ( - 16384)
230  IF X < 128 THEN 390
240 :
250  REM PRESSION D'UNE TOUCHE
260 :
270 X = X - 128
280  IF X = 21 THEN CP = CP / 2
290  IF X = 8 THEN CP = CP * 2
300  IF CP < 0.1 THEN CP = 0.1
310  IF CP > 4 THEN CP = 4
320 :
330  REM CLAVIER REINITIALISE
340 :
350  POKE  - 16368,0
360 :
370  REM  BOUCLE DE TEMPORISATION
380 :
390  FOR P = 1 TO DP * CP: NEXT P
400  RETURN
410 :
420  REM  /// BRUITAGE ///
430 :
440  FOR T = A TO B STEP S: POKE 768,T
     : POKE 769,H: POKE 796,1: CALL 7
     70: NEXT T
450  RETURN
460 :
470  REM  /// EXTRACTION ///
480 :
490  REM ### TRAITEMENT ###
500 :
510 X$ = PR$(I)
520 :
530  REM ### ANIMATION ###
540 :
550 M = 1: GOSUB 100
560 DP = 500: GOSUB 160
570 :
580  REM  --- NUMERO EN INVERSE ---
590 :
600  VTAB 3 + 2 * I: INVERSE : PRINT I
     ;
610 A = 1:B = 5:S = 1:H = 200: GOSUB 4
     20
620 :
630  REM  --- GLISSEMENT 1 ---
640 :
650 DP = 500: GOSUB 160
660 LX =  LEN (X$)
670  FOR K = 0 TO LX
680  NORMAL
690  HTAB 4: PRINT  MID$ (X$,1,K);
700  INVERSE : PRINT X$;
710 A = 1:B = 5:S = 1:H = 40: GOSUB 42
     0
720  NEXT K
730 :
740  REM  --- GLISSEMENT 2 ---
750 :
760  FOR K = 1 TO 22 - LX - 4
770  NORMAL : HTAB LX + 4: PRINT  SPC(
      K)
780  INVERSE : PRINT X$;
790 A = 1:B = 5:S = 1:H = 40: GOSUB 42
     0
800  NEXT K
810 :
820  REM  --- ARRET A DROITE ---
830 :
840 DP = 500: GOSUB 160
850  RETURN
860 :
870  REM  /// COMPARAISON ///
880 :
890  REM  ### GLISSEMENT VERS LE HAUT
     ###
900 :
910  FOR L = 5 TO 4 STEP  - 1
920  VTAB L + 2 * J: HTAB 22: CALL  -
     868
930  INVERSE
940  VTAB L - 1 + 2 * J: HTAB 22: PRIN
     T X$
950 A = 10:B = 1:S =  - 1:H = 10: GOSU
     B 420
960  NEXT L
970 :
980  REM  ELEMENT COMPARE EN FLASH
990 :
1000 M = 2: GOSUB 100
1010 DP = 100: GOSUB 160
1020  FLASH
1030  VTAB 3 + 2 * J: HTAB 4: PRINT PR
      $(J)
1040  FOR F = 1 TO 2
1050 A = 20:B = 10:S =  - 1:H = 20: GO
     SUB 420
1060 A = 10:B = 20:S = 1:H = 20: GOSUB
      420
1070  NEXT F
1080  NORMAL
1090  VTAB 3 + 2 * J: HTAB 4: PRINT PR
      $(J)
1100 DP = 100: GOSUB 230
1110  RETURN
1120 :
1130  REM  /// DECALAGE ///
1140 :
1150  REM  ### TRAITEMENT ###
1160 :
1170 PR$(J + 1) = PR$(J)
1180 :
1190  REM  ### ANIMATION ###
1200 :
1210  REM  --- MESSAGE BAS ECRAN ---
1220 :
1230 M = 3: GOSUB 100
1240 DP = 500: GOSUB 160
1250 :
1260  REM   AFFICHAGE DU DECALAGE
1270 :
1280 A = 10:B = 1:S =  - 1:H = 80: GOS
     UB 420
1290  INVERSE : VTAB 4 + 2 * J: HTAB 4
     : PRINT PR$(J + 1)
1300 DP = 150: GOSUB 160
1310  VTAB 4 + 2 * J: HTAB 4: NORMAL :
      PRINT  SPC( 20)
1320  VTAB 5 + 2 * J: HTAB 4: INVERSE
     : PRINT PR$(J + 1);: NORMAL : PR
     INT  SPC( 20)
1330 A = 10:B = 1:S =  - 1:H = 80: GOS
     UB 420
1340 DP = 1000: GOSUB 160
1350  VTAB 5 + 2 * J: HTAB 4: PRINT PR
      $(J + 1)
1360  RETURN
1370 :
1380  REM  /// INSERTION ///
1390 :
1400  REM  ### ANIMATION ###
1410 :
1420  REM  --- MESSAGE BAS ECRAN ---
1430 :
1440 M = 4: GOSUB 100
1450 :
1460  REM   GLISSEMENT VERS LE BAS
1470 :
1480 DP = 75: GOSUB 160
1490  VTAB 3 + 2 * J: HTAB 22: CALL  -
      868
1500  INVERSE : VTAB 4 + 2 * J: PRINT
     X$
1510 A = 15:B = 1:S =  - 1:H = 80: GOS
     UB 420
1520 DP = 75: GOSUB 160
1530  VTAB 4 + 2 * J: HTAB 22: CALL  -
      868
1540  VTAB 5 + 2 * J: PRINT X$;
1550 A = 15:B = 1:S =  - 1:H = 80: GOS
     UB 420
```

```
1560 DP = 1000: GOSUB 160
1570 :
1580  REM   GLISSEMENT HORIZONTAL
1590 :
1600  FOR L = 21 TO 4 STEP  - 1
1610  INVERSE : HTAB L: PRINT XS;
1620  NORMAL : PRINT  SPC( 1)
1630 A = L:B = L:S = 1:H = 100: GOSUB
     420
1640  NEXT L
1650 :
1660  REM  INSERTION TERMINEE
1670 :
1680 DP = 1000: GOSUB 160
1690  NORMAL : HTAB 4: PRINT XS
1700 :
1710  REM   TRAITEMENT
1720 :
1730 PRS(J + 1) = XS
1740 J = 0
1750  RETURN
1760 :
1770  REM  TRI PAR INSERTION
1780 :
1790  FOR I = 2 TO N
1800  GOSUB 470
1810  FOR J = I - 1 TO 0 STEP  - 1
1820  GOSUB 870
1830  ON (X$ < PR$(J)) + 1 GOSUB 1380,
     1130
1840  NEXT J,I
1850  RETURN
1860 :
1870  REM  /// FIN DU PROGRAMME ///
1880 :
1890  REM  MESSAGE BAS DE L'ECRAN
1900 :
1910 M = 5: GOSUB 100
1920 A = 1:B = 10:S = 1:H = 100: GOSUB
     420
1930 A = 1:B = 10:S = 1:H = 50: GOSUB
     420
1940 :
1950 : REM   ATTENTE D'UNE TOUCHE
1960 :
1970  POKE 49168,0: WAIT 49152,128: PO
     KE 49168,0
1980 :
1990  REM  TEST RETOUR TRI.CHAINE.MENU
2000 :
2010 TC =  PEEK (49152)
2020 CH = (TC = 27) + (TC = 13) * 2
2030  ON CH GOSUB 2060,2110
2040  GOTO 1950
2050 :
2060  REM  /// SORTIE ///
2070 :
2080  TEXT : HOME
2090  POP : RETURN
2100 :
2110  REM /// TRI.CHAINE.MENU ///
2120 :
2130  POP : POP
2140  TEXT : VTAB 1
2150 PRO =  PEEK (48896) = 76
2160  ON 2 - PRO GOTO 2170,2180
2170  PRINT  CHR$ (4);"RUN BASIC/TRI.C
     HAINE.MENU"
2180  PRINT  CHR$ (4);"RUN TRI.CHAINE.
     MENU"
2190 :
2200  REM  /// INITIALISATION ///
2210 :
2220  PRINT  CHR$ (21)
2230  TEXT : HOME : SPEED= 255
2240 :
2250  REM  --- TABLE DES PAYS ---
2260 :
2270  DIM PRS(9)
2280 N =  - 1
2290 :
2300  REM  --- LECTURE PAYS --
2310 :
2320  READ PRS
2330  IF PR$ = "F" THEN 2370
2340 N = N + 1:PRS(N) = PRS
2350  GOTO 2300
2360 :
2370  REM  LECTURE DES MESSAGES
2380 :
2390  DIM M$(5)
2400  FOR K = 0 TO 5
2410  READ M$(K)
2420  NEXT K
2430 :
2440  REM  --- COEFFICIENT PAUSE ---
2450 :
2460 CP = 1
2470 :
2480  REM  BRUITAGE
2490 :
2500  FOR I = 0 TO 25: READ A: POKE 77
     0 + I,A: NEXT I
2510 :
2520  REM  --- AFFICHAGE INITIAL ---
2530 :
2540  REM  --- CADRE ---
2550 :
2560  POKE 48,186
2570  VLIN 0,47 AT 0: VLIN 0,47 AT 39
2580 LS = "::::::::::::::::::::::::::::
     :::::::::::::"
2590  VTAB 2: PRINT LS;
2600  VTAB 22: PRINT LS;
2610  VTAB 24: PRINT  LEFT$ (LS,39);
2620  POKE 33,37: POKE 32,2
2630 :
2640  REM  --- TITRE ---
2650 :
2660  VTAB 1
2670 A$ = "T R I  P A R  I N S E R T I
     O N"
2680 A = 19 -  LEN (A$) / 2
2690  HTAB A: PRINT A$
2700  VTAB 3
2710 :
2720  REM  AFFICHAGE DES PAYS
2730 :
2740  FOR K = 0 TO N
2750  PRINT K;". ";PR$(K): PRINT
2760  NEXT K
2770 :
2780  REM MESSAGE BAS DE L 'ECRAN
2790 :
2800 M = 0: GOSUB 100
2810 A = 1:B = 10:S = 1:H = 100: GOSUB
     420
2820 A = 1:B = 10:S = 1:H = 50: GOSUB
     420
2830 :
2840  REM ATTENTE D'UNE TOUCHE
2850 :
2860  WAIT  - 16384,128: POKE  - 16368
     ,0
2870  RETURN
2880 :
2890  REM  ### DONNEES FIXES ###
2900 :
2910  REM  --- PAYS ---
2920 :
2930  DATA " "
2940 :
2950  DATA  "PASCAL"
2960  DATA  "FORTRAN"
2970  DATA  "COBOL"
2980  DATA  "BASIC"
2990  DATA  "ASSEMBLEUR"
3000  DATA  "MODULA 2"
3010  DATA  "ADA"
3020  DATA  "PL1"
3030  DATA  "C"
3040  DATA  "F"
3050 :
3060  REM  --- MESSAGES ---
3070 :
3080  DATA "D E B U T  D U  T R I"
3090  DATA "E X T R A C T I O N"
3100  DATA "C O M P A R A I S O N"
3110  DATA "D E C A L A G E"
3120  DATA "I N S E R T I O N"
3130  DATA "FIN: <ESC> = SORTIE <RTN>
     = MENU"
3140 :
3150  REM  --- PROG. MUSIQUE ---
3160 :
3170  DATA 173,48,192,173,0,3,32,168,2
     52,173,1,3,208,5,206,28,3,240,6,
     206,1,3,76,2,3,96
```

Source 'TRI.PRO.INIT'
assembleur Procode

```
 1          LST OFF
 2          TAB 03,15,21,31,36,49
 3 ********************************************
 4 *              >>> TRI.PRO.INIT <<<        *
 5 *------------------------------------------*
 6 * - réserve 1 mémoire-tampon d'une page entre *
 7 *   le BASIC.SYSTEM et sa mémoire-tampon    *
 8 *   d'usage général (General Purpose Buffer). *
 9 * - le programme TRI.CHAINE.C y est chargé.  *
10 *------------------------------------------*
11 * - ProDOS + BASIC.SYSTEM + Applesoft       *
12 *------------------------------------------*
13 * - Auteur:            LAMBILLON Bernard    *
14 * - Date de création: 12 aout 1985          *
15 * - Assembleur:        Merlin-Pro 1.43      *
16 ********************************************
17
18          ORG $300
19
20 *-------------------------------------------*
21 * - Adresses de référence                   *
22 *-------------------------------------------*
23
24 IN        =   $200        ;Mémoire-tampon d'entrée
25 WARMST    =   S3D0        ;Redémarrage avec SED
26 AMPERVB   =   S3F6        ;Pointeur perluète (&)
27 AMPERVH   =   S3F7        ;(vecteur Ampersand)
28 GETBUFR   EQU $BEF5       ;Allocation de mémoire par
29                           ;page de 256 octets; le
```

```
30                          ;nombre de pages est dans A.
31                          ;L'adresse de début de la page
32                          ;est un multiple de 256 (adresse
33                          ;base = $00).
34                          ;Au retour, A = 0 si erreur.
35 DOSCMD     EQU $BE03     ;Analyse et exécution d'une
36                          ;commande ProDOS plaçée dans
37                          ;la mémoire-tampon d'entrée;
38                          ;la ligne de commande se termine
39                          ;par $8D, le code ASCII des
40                          ;car. est avec le bit 7 mis à 1.
41                          ;Au retour, C = 1 si erreur.
42 ERROUT     EQU $BE09     ;Gestion en cas d'erreur
43                          ;dans une commande ProDos.
44                          ;Le code d'erreur est placé en
45                          ;ERRCODE ($BE0F) et ERRNUM ($DE)
46 PRINTERR   EQU $BE0C     ;Affiche message d'erreur
47                          ;dont le numéro est en A.
48 ERRCODE    =   $BE0F     ;Code d'une erreur ProDOS
49
50 *-----------------------------------------*
51 * - réservation d'une page de 256 octets  *
52 *-----------------------------------------*
53
54          LDA #$01        ;1 page de 256 octets
55          JSR GETBUFR     ;allocation de mémoire
56          BEQ AMPER       ;si pas d'erreur
57          PAG
58 *-----------------------------------------*
59 *- Erreur: plus assez de place en mémoire *
60 *-----------------------------------------*
61
62          LDA #$0E        ;Code erreur: PROGRAM TOO LARGE
63          JSR PRINTERR    ;affiche message erreur
64          JMP WARMST      ;arret du programme Basic
65
66 *-----------------------------------------*
67 * - Ecriture(partie haute du vecteur AMPERV *
68 *-----------------------------------------*
69
70 AMPER      STA AMPERVH   ;AMPERVH = n° de page
71                          ;de la zone allouée
72
73 *-----------------------------------------*
74 * - conversion de l'adresse en ASCII      *
75 *   et écriture de celle-ci dans la zone  *
76 *   ADRESSE qui mémorise l'adresse de     *
77 *   chargement du programme TRI.CHAINE    *
78 *-----------------------------------------*
79
80          TAX             ;sauvegarde de A dans X
81          LSR             ;décalage du quartet de
82          LSR             ;poids fort dans le
83          LSR             ;quartet de poids faible
84          LSR             ;ex: 9C --> 09
85          CLC             ;conversion en Ascii
86          ADC #$B0        ;ex: 09 --> B9
87          STA ADRESSE     ;paramètre adresse de chargem.
88
89          TXA             ;restauration de A
90          AND #$0F        ;masque $0F (ex: $9C--> $0C)
91                          ;A = quartet de poids faible
92          CMP #$0A        ;$00 à $09 ou $0A à $0F
93          BMI CHIFFRE     ;si A contient de $00 à $09
94          ADC #$06        ;sinon A <== A + 7
95                          ;ex: $0C --> $13
96 CHIFFRE  ADC #$B0        ;conversion en Ascii
97                          ;ex: $13 --> $C3 (lettre C)
98          STA ADRESSE+1   ;paramètre adresse de chargem.
99
100 *-----------------------------------------*
101 * - Ecriture partie basse du vecteur AMPERV *
102 *-----------------------------------------*
103
104         LDA #$00        ;alignement sur 1 page
105         STA AMPERVB
106
107 *-----------------------------------------*
108 * - Copie, analyse et exécution de la     *
109 *   commande ProDOS: BLOAD TRI.CHAINE.    *
110 *-----------------------------------------*
111
112         LDY #0          ;copie de la commande
113 COPIE   LDA COMMANDE,Y  ;dans la mémoire-tampon
114         STA IN,Y        ;d'entrée IN ($200).
115         INY
116         CMP #$8D        ;fin de commande?
117         BNE COPIE
118         JSR DOSCMD      ;analyse et exécution
119         BCC RETOUR      ;si pas d'erreur
120         PAG
121 *-----------------------------------------*
122 * - Erreur d'exécution de la commande     *
123 *-----------------------------------------*
124
125         JSR ERROUT      ;analyse l'erreur
126         LDA ERRCODE     ;imprime le message
127         JSR PRINTERR    ;affiche message erreur
128         JMP WARMST      ;arret du programme Basic
129
130 RETOUR  RTS             ;retour au programme Basic
131
132 *-----------------------------------------*
133 * - Commande:  BLOAD TRI.CHAINE.C, A$??00 *
134 *-----------------------------------------*
135
136 COMMANDE  ASC "BLOADTRI.CHAINE.C,A$"
137 ADRESSE   DFB $B0,$B0,$B0,$B0
138         DFB $8D         ;fin de la commande
```

Source 'TRI.DOS.INIT'
assembleur Procode

```
1           LST OFF
2           TAB 03,15,21,31,36,49
3 ******************************************
4 *            >>> TRI.DOS.INIT <<<        *
5 *----------------------------------------*
6 * - Ce programme charge le programme     *
7 *   TRI.CHAINE.C en dessous du HIMEM actuel. *
8 * - le vecteur 6 est mis en place et le HIMEM *
9 *   est déplacé de 214 octets ($D6).     *
10 *----------------------------------------*
11 * - DOS 3.3 + Applesoft                 *
12 *----------------------------------------*
13 * - Auteur:          LAMBILLON Bernard  *
14 * - Date de création: 12 aout 1985      *
15 * - Assembleur:      Merlin-Pro 1.43    *
16 ******************************************
17
18          ORG $300
19
20 *----------------------------------------*
21 * - Adresses de référence               -*
22 *----------------------------------------*
23
24 LINNUMB  =   $50         ;adresse pour RSHM
25 LINNUMH  =   $51
26 HIMEMB   =   $73         ;adresse la plus élevée
27 HIMEMH   =   $74         ;de la mémoire vive
28 IN       =   $200        ;mémoire-tampon d'entrée
29 AMPERVB  =   $3F6        ;pointeur perluète (6)
30 AMPERVH  =   $3F7        ;(vecteur Ampersand)
31 DOSKBD   EQU $9FCD       ;analyse une commande DOS
32 RSHM     EQU $F28C       ;mise en place du HIMEM
33                          ;en fonction de LINNUM ($50,$51)
34
35 *----------------------------------------*
36 * - HIMEM <== HIMEM - $D6  (214 octets)  *
37 * - AMPERV <== HIMEM                     *
38 *----------------------------------------*
39
40          LDY #$03        ;init pour écriture ADRESSE
41          SEC
42          LDA HIMEMB      ;HIMEMB <== HIMEMB - $D6
43          SBC #$D7
44          STA LINNUMB     ;pour RSHM
45          STA AMPERVB     ;AMPERVB <== HIMEMB
46          JSR ASCII       ;conversion en Ascii
```

```
          LDA HIMEMH
          SBC #$00          ;HIMEMH <== HIMEMH - retenue
          STA LINNUMH       ;pour RSHM
          STA AMPERVH       ;AMPERVH <== HIMEMH
          JSR ASCII         ;conversion en Ascii
          JSR RSHM          ;ReSet HiMem

*-----------------------------------------------*
* - Copie, analyse et exécution de la          *
*   commande ProDOS: BLOAD TRI.CHAINE.         *
*-----------------------------------------------*

COPIE     LDA COMMANDE,Y    ;dans la mémoire-tampon
          STA IN,Y          ;d'entrée IN ($200).
          INY
          CMP #$8D          ;fin de commande?
          BNE COPIE
          JSR DOSKBD        ;analyse et exécution
          RTS

*-----------------------------------------------*
* - conversion de l'adresse en ASCII           *
*   et écriture de celle-ci dans la zone       *
*   ADRESSE qui mémorise l'adresse de          *
*   chargement du programme TRI.CHAINE         *
*-----------------------------------------------*

ASCII     TAX               ;sauvegarde de A dans X
          AND #$0F          ;masquage des bits 4 à 7
          JSR CONVERT       ;conversion quartet
          TXA
          LSR               ;décalage du quartet de
          LSR               ;poids fort dans le
          LSR               ;quartet de poids faible
          LSR               ;ex: 9C --> 09
          JSR CONVERT       ;conversion quartet de
                            ;poids fort
          RTS               ;fin de la routine ASCII

CONVERT   CLC
          CMP #$0A          ;$00 à $09 ou $0A à $0F
          BMI CONVERT1      ;si A contient de $00 à $09
          ADC #$06          ;sinon A <== A + 7
                            ;ex: $0C --> $13
CONVERT1  ADC #$B0          ;conversion en Ascii
                            ;ex: $13 --> $C3 (lettre C)
          STA ADRESSE,Y     ;paramétre adresse de chargem.
          DEY
          RTS               ;fin de la routine CONVERT

*-----------------------------------------------*
* - Commande:  BLOAD TRI.CHAINE.C, A$??00      *
*-----------------------------------------------*

COMMANDE  ASC "BLOADTRI.CHAINE.C,A$"
ADRESSE   DS  4
          DFB $8D           ;fin de la commande
```

Source 'TRI.CHAINE'
assembleur Procode

```
          LST OFF
          TAB 03,15,21,31,36,49
*************************************************
*            >>> TRI.CHAINE <<<                 *
*-----------------------------------------------*
* - trie un tableau de chaines de caractères   *
*   sans modifier l'ordre initial.             *
* - mémorise l'ordre alphabétique au moyen     *
*   d'une table d'index (type entier) qui est  *
*   initialisée par le programme lui-meme.     *
*-----------------------------------------------*
* - Syntaxe : & N, A$(1), IX%(1)               *
*   avec N  : nombre d'éléments à trier        *
*      A$(1): élément 1 de la table à trier    *
*     IX%(1): élément 1 de la table d'index    *
*-----------------------------------------------*
* - DOS + Applesoft      ou                    *
* - ProDOS + Basic.System + Applesoft          *
*-----------------------------------------------*
* - Auteur          : LAMBILLON Bernard        *
* - Date de création: 10 aout 1985             *
* - Assembleur      : ProCode 1.23             *
*************************************************

*************************************************
* ADRESSES DE REFERENCE                        *
*************************************************
```

Comment faire ?

Pour utiliser cette routine de tri au sein de votre programme Basic, vous devrez avoir sur la disquette les fichiers binaires :

`TRI.DOS.INIT.C  et  TRI.CHAINE.C`

s'il s'agit d'une disquette DOS, ou :

`TRI.PRO.INIT.C  et  TRI.CHAINE.C`

s'il s'agit d'une disquette ProDOS.

Au début du programme Basic, il y aura une instruction :

`PRINT  CHR$(4)"BRUN  TRI.DOS.INIT.C"`

s'il s'agit d'une disquette DOS, ou :

`PRINT  CHR$(4)"BRUN  TRI.PRO.INIT.C"`

s'il s'agit d'une disquette ProDOS.

Comment trier ?

Un exemple simple :

Dans un tableau alphanumérique se trouvent des noms de clients :

```
DIM A$(6) : A$(1) = "JEAN" : A$(2) =
  "PAUL" : A$(3) = "ANNE" : A$(4) =
  "YVES" : A$(5) = "LYDA" : A$(6) =
                "YANN"
```

Il faut dimensionner un tableau pour recevoir les clefs d'accès :

`DIM  I%(6)`

Maintenant le tri proprement dit :

`&6, A$(1), I%(1)`

&6 parce qu'il y a 6 éléments à trier,
A$(1) représente le premier élément à trier,
I%(1) est le premier index.

Pour trouver nos clients par ordre alphabétique, il faut les lire suivant le tableau d'index maintenant modifié :

```
FOR I = 1 TO 6 : PRINT A$(I%(I)) :
                NEXT
```

Les programmes de démonstration

sous DOS fonctionneront directement et d'autant plus simplement que vous disposez de la disquette d'accompagnement.

Sous ProDOS, après saisie au clavier ou transfert depuis la disquette Pom's, vous devrez mettre les programmes Basic dans un sous-catalogue nommé :

`BASIC`

et les routines dans un sous-catalogue nommé :

`CODE.MACHINE`

```
28
29 *-----------------------------------------*
30 * Pointeurs vers la table des chaines à trier *
31 *-----------------------------------------*
32
33 TABLE0B   =    $6        ;adresse de l'élément
34 TABLE0H   =    $7        ;d'indice 0 de la table
35 TABLE1B   =    $8        ;adresse de l'élément
36 TABLE1H   =    $9        ;extrait de la table
37 TABLE2B   =    $19       ;adresse de l'élément
38 TABLE2H   =    $1A       ;comparé dans la table
39
40 *-----------------------------------------*
41 * Pointeurs vers la table d'index          *
42 *-----------------------------------------*
43
44 INDEX1B   =    $83       ;adresse de l'index
45 INDEX1H   =    $84       ;de l'élément extrait
46 INDEX2B   =    $1B       ;adresse de l'index
47 INDEX2H   =    $1C       ;de l'élément comparé
48
49 *-----------------------------------------*
50 * Nombre d'éléments à trier                *
51 *-----------------------------------------*
52
53 NBELB     =    $50       ;la routine GETADR écrit un
54 NBELH     =    $51       ;entiér à cette adresse
55
56 *-----------------------------------------*
57 * Descripteur de la chaine extraite        *
58 *-----------------------------------------*
59
60 LONG1     =    $93       ;longueur chaine extraite
61 CHAINE1B  =    $94       ;adresse de stockage
62 CHAINE1H  =    $95       ;de la chaine extraite
63          PAG
64 *-----------------------------------------*
65 * Descripteur de la chaine comparée        *
66 *-----------------------------------------*
67
68 LONG2     =    $96       ;longueur chaine comparée
69 CHAINE2B  =    $97       ;adresse de stockage
70 CHAINE2H  =    $98       ;de la chaine comparée
71
72 *-----------------------------------------*
73 * Index de l'élément extrait               *
74 *-----------------------------------------*
75
76 ENTIER1H  =    $9A       ;partie haute de l'index
77 ENTIER1B  =    $9B       ;partie basse de l'index
78
79 *-----------------------------------------*
80 * Index de l'élément comparé               *
81 *-----------------------------------------*
82
83 ENTIER2H  =    $1D       ;partie haute de l'index
84 ENTIER2B  =    $1E       ;partie basse de l'index
85
86 *-----------------------------------------*
87 * RoutInes Applesoft en MEM (ROM)          *
88 *-----------------------------------------*
89
90 FRMNUM    EQU  $DD67     ;value la formule pointé;
91                          ;par TXTPTR ($B8 et $B9)
92                          ;et place le résultat en
93                          ;FAC ($9D..$A2)
94 CHKCOM    EQU  $DEBE     ;teste présence d'une virgule
95                          ;si erreur: ?SYNTAX ERROR
96                          ;IN numéro de ligne
97                          ;et retour au programme Basic
98 CHKNUM    EQU  $DD6A     ;teste variable type numérique
99                          ;si erreur: ?TYPE MISMATCH
100                         ;ERROR IN numéro de ligne
101 CHKSTR   EQU  $DD6C     ;teste variable type chaine
102                         ;si erreur: ?TYPE MISMATCH
103                         ;ERROR IN numéro de ligne
104 PTRGET   EQU  $DFE3     ;cherche une variable et place
105                         ;son adresse dans A et Y et en
106                         ;VARPNT ($83 et $84)
107 GETADR   EQU  $E752     ;transforme de FAC ($9D..$A2)
108                         ;en un entier stocké en
109                         ;LINNUM ($50,$51) c-à-d NBLE
110                         ;(nombre d'éléments à trier)
111
112 ***********************************************
113 * INITIALISATION DU TRI                        *
114 ***********************************************
115
116 *-----------------------------------------*
117 * Lecture du nombre d'éléments à trier (NBLE) *
118 *-----------------------------------------*
119
120          JSR  FRMNUM    ;lecture du nombre d'éléments;
121                         ;et stockage en FAC ($9D..$A2)
122          JSR  GETADR    ;FAC --> NBLE ($50 et $51)
123          PAG
124 *-----------------------------------------*
125 * Initialisation pointeurs TABLE0 et TABLE1   *
126 *-----------------------------------------*
127
128          JSR  CHKCOM    ;teste présence d'une virgule
129          JSR  PTRGET    ;place adresse de la variable
130                         ;pointée par TXTPTR dans A
131                         ;(partie basse) et Y (partie
132                         ;haute)
133          JSR  CHKSTR    ;teste variable type chaine
134          SEC            ;initiation soustraction
135          STA  TABLE1B   ;transfert de A --> TABLE1B
136          SBC  #$03      ;3 octets par élément d'un
137                         ;tableau du type chaine
138          STA  TABLE0B   ;transfert de A --> TABLE0B
139          STY  TABLE1H   ;transfert de Y --> TABLE1H
140          TYA
141          SBC  #$00      ;TABLE0H <-- TABLE1H - carry
142          STA  TABLE0H   ;transfert de A --> TABLE0H
143
144 *-----------------------------------------*
145 * Initialisation du pointeur INDEX1            *
146 *-----------------------------------------*
147
148          JSR  CHKCOM    ;teste présence d'une virgule
149          JSR  PTRGET    ;place adresse de l'élément 1
150                         ;de la table d'index en VARPNT
151                         ;($83 et $84) c-à-d ici INDEX1
152          JSR  CHKNUM    ;teste s'il s'agit d'un nombre
153
154 *-----------------------------------------*
155 * Initialisation compteur ENTIER1 et IX%(1)    *
156 *-----------------------------------------*
157
158          LDY  #$00      ;ENTIER1H <== 0
159          STY  ENTIER1H
160          TYA
161          STA  (INDEX1B),Y;partie haute IX%(1) <== 0
162          INY
163          STY  ENTIER1B  ;ENTIER1B <== 1
164          TYA
165          STA  (INDEX1B),Y;partie basse IX%(1) <== 1
166          CLC            ;init. report pour EXTRACT
167
168 ***********************************************
169 * BOUCLE EXTERNE (FOR I = 2 TO N)              *
170 ***********************************************
171
172 *-----------------------------------------*
173 * Répéter NBEL <== NBEL-1 tant que NBEL <> 0   *
174 *-----------------------------------------*
175
176 EXTERNE  LDA  NBELB
177          BNE  EXTERNE1  ;si NBELB<>0
178          DEC  NBELH     ;NBELH <== NBELH-1 si NBLEB=0
179 EXTERNE1 DEC  NBELB     ;NBELB <== NBELB-1
180          BNE  EXTRACT   ;si NBLEB<>0
181          LDA  NBELH
182          BNE  EXTRACT   ;si NBLEB=0 et NBELH<>0
183          RTS            ;si NBELB=0 et NBELH=0
184          PAG
185 ***********************************************
186 * INSERTION                                    *
187 ***********************************************
188
189 *-----------------------------------------*
```

```
190 * ENTIER1 <== ENTIER1 + 1                          *
191 *-----------------------------------------------*
192
193 INSERT    INC  ENTIER1B  ;ENTIER1B <== ENTIER1B+1
194           BNE  INSERT1   ;si pas de chgt. de page
195           INC  ENTIER1H  ;ENTIER1H <== ENTIER1H+1
196
197 *-----------------------------------------------*
198 * Copie de ENTIER1 dans l'élément pointé par    *
199 * INDEX2 + 2                                    *
200 *-----------------------------------------------*
201
202 INSERT1   LDA  ENTIER1H  ;copie partie haute
203           LDY  #$02
204           STA  (INDEX2B),Y
205           LDA  ENTIER1B  ;copie partie basse
206           INY
207           STA  (INDEX2B),Y
208           CLC            ;bit de report forcé à 0
209                          ;init. report pour EXTRACT
210           BCC  EXTERNE   ;branchement inconditionnel
211
212 ***********************************************
213 * EXTRACTION                                   *
214 ***********************************************
215
216 *-----------------------------------------------*
217 * TABLE1 <-- TABLE1 + 3                        *
218 *-----------------------------------------------*
219
220 EXTRACT   LDA  TABLE1B   ;TABLE1B <== TABLE1B+3
221           ADC  #$03
222           STA  TABLE1B
223           BCC  EXTRACT1  ;si pas de report
224           INC  TABLE1H   ;TABLE1H <== TABLE1H+3
225
226 *-----------------------------------------------*
227 * INDEX2 <== INDEX1                            *
228 * INDEX1 <== INDEX1 + 2                        *
229 *-----------------------------------------------*
230
231           CLC
232 EXTRACT1  LDA  INDEX1B   ;INDEX2B <== INDEX1B
233           STA  INDEX2B
234           ADC  #$02      ;INDEX2B <== INDEX1B+2
235           STA  INDEX1B
236           LDA  INDEX1H   ;INDEX2H <== INDEX1H
237           STA  INDEX2H
238           ADC  #$00      ;INDEX1H <== INDEX1H+report
239           STA  INDEX1H
240           PAG
241
242 *-----------------------------------------------*
243 * Sauvegarde du descripteur de chaine contenu   *
244 * dans l'élément extrait.                       *
245 *-----------------------------------------------*
246
247           LDY  #02       ;copie de 3 octets
248 EXTRACT2  LDA  (TABLE1B),Y
249           STA  LONG1,Y   ;écriture en CHAINE1H,
250           DEY            ;CHAINE1B et LONG1
251           BPL  EXTRACT2
252
253 ***********************************************
254 * BOUCLE INTERNE (FOR J=I-1 TO 0 STEP -1)      *
255 ***********************************************
256
257 *-----------------------------------------------*
258 * TABLE2 <== 3 * ENTIER2 + TABLE0              *
259 *-----------------------------------------------*
260
261 INTERNE   LDY  #$00
262           LDA  (INDEX2B),Y ;lecture de ENTIER2H
263           STA  TABLE2H   ;copie de ENTIER2H en TABLE2H
264           STA  ENTIER2H  ;et sauvegarde en ENTIER2H
265           INY
266           LDA  (INDEX2B),Y ;lecture de ENTIER2B
267           STA  TABLE2B   ;copie de ENTIER2B en TABLE2B
268           STA  ENTIER2B  ;et sauvegarde en ENTIER2B
269           ASL            ;A contient 2*ENTIER2B;
270           ROL  TABLE2H   ;TABLE2H <== 2*ENTIER2H+report
271           ADC  TABLE2B   ;A contient 3*ENTIER2B
272           BCC  INTERNE1  ;si pas de débordement
273           INC  TABLE2H   ;TABLE2H <== TABLE2H+1
274           CLC
275 INTERNE1  ADC  TABLE0B   ;A contient 3*ENTIER2B+TABLE0B
276           STA  TABLE2B   ;ok pour TABLE2B
277           LDA  TABLE2H   ;A contient 2*ENTIER2H+report
278           ADC  ENTIER2H  ;A contient 3*ENTIER2H
279           CLC
280           ADC  TABLE0H   ;A contient 3*ENTIER2H+TABLE0H
281           STA  TABLE2H   ;ok pour TABLE2H
282
283 *-----------------------------------------------*
284 * Sauvegarde du descripteur de chaine contenu   *
285 * dans l'élément comparé.                       *
286 *-----------------------------------------------*
287
288           LDY  #$02      ;copie de 3 octets
289 INTERNE2  LDA  (TABLE2B),Y
290           STA  LONG2,Y   ;écriture en CHAINE2H,
291           DEY            ;CHAINE2B et LONG2
292           BPL  INTERNE2
293           PAG
294 ***********************************************
295 * COMPARAISON                                   *
296 ***********************************************
297
298 *-----------------------------------------------*
299 * Test dernier caractère d'une des chaines?     *
300 * Test si une des chaines est vide?             *
301 *-----------------------------------------------*
302
303 INTERNE3  INY            ;Y=0,1,2...
304           CPY  LONG2
305           BEQ  INSERT    ;si dernier caractère chaine
306                          ;comparée (LONG2=0 ou LONG2=Y)
307           CPY  LONG1
308           BEQ  DECALE    ;si dernier caractère chainee
309                          ;extraite (LONG1=0 ou LONG1=Y)
310 *-----------------------------------------------*
311 * Comparaison du Yème caractère des 2 chaines *
312 *-----------------------------------------------*
313
314           LDA  (CHAINE2B),Y;A=Yème car. chaine comparée
315           CMP  (CHAINE1B),Y;M=Yème car. chaine extraite
316           BCC  INSERT    ;A < M
317           BNE  DECALE    ;A > M
318
319 *-----------------------------------------------*
320 * Caractères identiques                         *
321 *-----------------------------------------------*
322
323           BEQ  INTERNE3  ;branchement inconditionnel
324
325 ***********************************************
326 * DECALAGE                                      *
327 ***********************************************
328
329 *-----------------------------------------------*
330 * IX%(J+1) <== IX%(J)                           *
331 *-----------------------------------------------*
332
333 DECALE    LDY  #$02
334           LDA  ENTIER2H
335           STA  (INDEX2B),Y
336           INY
337           LDA  ENTIER2B
338           STA  (INDEX2B),Y
339
340 *-----------------------------------------------*
341 * NEXT J (boucle interne) , J <== J - 1         *
342 * INDEX2 <== INDEX2 - 2                         *
343 *-----------------------------------------------*
344
345           SEC
346           LDA  INDEX2B   ;INDEX2B <== INDEX2B-2
347           SBC  #$02
348           STA  INDEX2B
349           BCS  INTERNE   ;si pas de retenue
350           DEC  INDEX2H   ;INDEX2H <== INDEX2H-1
351           BCC  INTERNE   ;branchement inconditionnel
```

Récapitulation 'TRI.CHAINE.C'

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE TRI.CHAINE.C,A$7000,L$D7

```
7000- 20 67 DD 20 52 E7 20 BE
7008- DE 20 E3 DF 20 6C DD 38
7010- 85 08 E9 03 85 06 84 09
7018- 98 E9 00 85 07 20 BE DE
7020- 20 E3 DF 20 6A DD A0 00
7028- 84 9A 98 91 83 C8 84 9B
7030- 98 91 83 18 A5 50 D0 02
7038- C6 51 C6 50 D0 19 A5 51
7040- D0 15 60 E6 9B D0 02 E6
7048- 9A A5 9A A0 02 91 1B A5
7050- 9B C8 91 1B 18 90 DD A5
7058- 08 69 03 85 08 90 03 E6
7060- 09 18 A5 83 85 1B 69 02
7068- 85 83 A5 84 85 1C 69 00
7070- 85 84 A0 02 B1 08 99 93
7078- 00 88 10 F8 A0 00 B1 1B
7080- 85 1A 85 1D C8 B1 1B 85
7088- 19 85 1E 0A 26 1A 65 19
7090- 90 03 E6 1A 18 65 06 85
7098- 19 A5 1A 65 1D 18 65 07
70A0- 85 1A A0 02 B1 19 99 96
70A8- 00 88 10 F8 C8 C4 96 F0
70B0- 92 C4 93 F0 0A B1 97 D1
70B8- 94 90 88 D0 02 F0 ED A0
70C0- 02 A5 1D 91 1B C8 A5 1E
70C8- 91 1B 38 A5 1B E9 02 85
70D0- 1B B0 A9 C6 1C 90 A5
```

Récapitulation 'TRI.PRO.INIT.C'

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE TRI.PRO.INIT.C,A$300,L$69

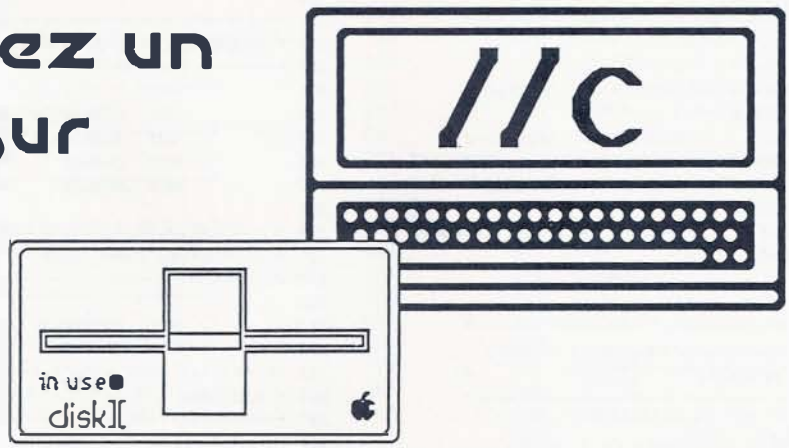
```
0300- A9 01 20 F5 BE F0 08 A9
0308- 0E 20 0C BE 4C D0 03 8D
0310- F7 03 AA 4A 4A 4A 4A 18
0318- 69 B0 8D 64 03 8A 29 0F
0320- C9 0A 30 03 18 69 07 69
0328- B0 8D 65 03 A9 00 8D F6
0330- 03 A0 00 B9 50 03 99 00
0338- 02 C8 C9 8D D0 F5 20 03
0340- BE 90 0C 20 09 BE AD 0F
0348- BE 20 0C BE 4C D0 03 60
0350- C2 CC CF C1 C4 D4 D2 C9
0358- AE C3 C8 C1 C9 CE C5 AE
0360- C3 AC C1 A4 B0 B0 B0 B0
0368- 8D
```

Récapitulation 'TRI.DOS.INIT.C'

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE TRI.DOS.INIT.C,A$300,L$63

```
0300- A0 03 38 A5 73 E9 D7 85
0308- 50 8D F6 03 20 2D 03 A5
0310- 74 E9 00 85 51 8D F7 03
0318- 20 2D 03 20 8C F2 B9 4A
0320- 03 99 00 02 C8 C9 80 D0
0328- F5 20 CD 9F 60 AA 29 0F
0330- 20 3C 03 8A 4A 4A 4A 4A
0338- 20 3C 03 60 18 C9 0A 30
0340- 02 69 06 69 B0 99 5E 03
0348- 88 60 C2 CC CF C1 C4 D4
0350- D2 C9 AE C3 C8 C1 C9 CE
0358- C5 AE C3 AC C1 A4 00 00
0360- 00 00 8D
```

Branchez un disk][sur le //c



Bruno Fénart

Suivant le principe cher à Apple de l'*incompatibilité minimale*, l'Apple //c ne pouvait utiliser que des lecteurs spécifiques, alors que la différence avec les disk][est minime, en dehors de la présentation. Comme bien souvent en informatique, le problème se résume à une question de connexion.

Le montage ci-dessous – peut-on parler de montage considérant la simplicité – a donc pour objet d'autoriser le branchement de votre lecteur d'Apple][+ ou //e sur votre nouveau //c. Avec un //c *équipé du kit Unidisk*, vous pourrez brancher votre disk][en série avec l'Unidisk 3.5 (se référer au manuel Unidisk 3.5).

Pour cela, il suffit de réaliser une liaison entre le câble plat du lecteur et le connecteur du lecteur de disquette qui se trouve sur la face arrière du //c.

Le montage

Il convient de se reporter au schéma de branchement car la différence des numérotations normalisées pour les connecteurs Berg et DB complique les choses. Attention, si ce montage est simple à réaliser, toute erreur risque d'être fatale. L'interface lecteur de disquettes de l'Apple //c est bien moins protégée que l'interface des][+ et //e, elle ne se relèverait pas d'un court-circuit. Le branchement entre le câble du lecteur et votre montage ne comporte pas de détrompeur ; veiller à ce que les câbles plats des deux connecteurs 2X10 partent dans des directions opposées.

Le manuel du DOS donne le schéma des circuits de l'interface et de la carte analogique du disk][(avec toutefois une erreur sur le signal RDDATA qui est inversé). Côté Apple //c, le manuel de référence donne les valeurs des signaux de chacune des broches du connecteur externe. Il suffisait de comparer ces tables et d'en déduire le branchement nécessaire.

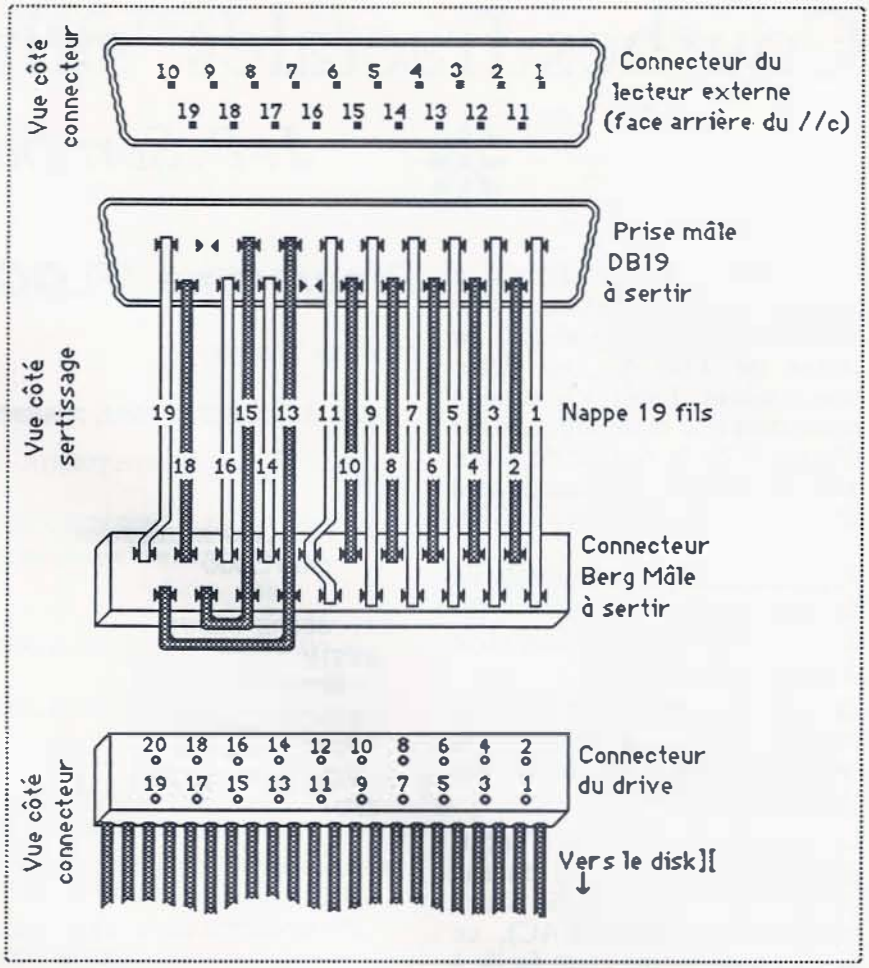
Matériel

Connecteur mâle type DB19 à sertir,
Connecteur mâle Berg 2X10 à sertir,
Câble plat 19 fils.

Connecteur DB19 (vers le //c)	Câble plat 19 fils	Connecteur BERG2X10
1	← 1 →	1
11	← 2 →	2
2	← 3 →	3
12	← 4 →	4
3	← 5 →	5
13	← 6 →	6
4	← 7 →	7
14	← 8 →	8
5	← 9 →	9
15	← 10 →	10
6	← 11 →	11.12
N.C.	← 12 →	N.C.
7	← 13 →	13.19
17	← 14 →	14
8	← 15 →	15.17
18	← 16 →	16
N.C.	← 17 →	N.C.
19	← 18 →	18
10	← 19 →	20

(N.C. = non connecté)

N.D.L.R. : Est-il nécessaire de préciser ici que ce type d'intervention sur votre matériel exclut tout recours à la garantie du constructeur en cas de problème ? Contrôler avec attention votre montage avec les tableaux de connexion des fils évitera les soucis...



Broche du //c	Signal	Description
1, 2, 3, 4	GND	Masse signal et alimentation
5	-12V	
6	+5V (Vcc)	
7,8	+12V	
9	EXTINT*	Interruption externe (non utilisée)
10	WRPROT	Écriture protégée (entrée)
11-14	SEEKPH0*	Phase 0 à 3 du moteur du bras
15	WRREQ	Demande d'écriture
16	-	non connecté
17	DR2* (DR1)	Sélection du disque 2 ou 1
18	RDDATA	Lecture des données (entrée)
19	WRDATA	Écriture des données (sortie)

(* = actif niveau bas)

PRINuu8i8uu@˜¶

Le laser a encore frappé !

La ligne 1010 du programme 'Analyse Multicritère' de Pom's 24 présentait en effet tous les symptômes d'une grande confusion. Il fallait lire :

```
...INVERSE : PRINT
C3;" "; : NORMAL :
INPUT " "; C3$ : IF
LEN (C3$)...
```

Courbes fractales, suite...

J.-P. Sorribas

Le numéro 22 de Pom's a présenté, pour les possesseurs de Macintosh, un thème abordé en classe de 1ère S. Les deux programmes listés ci-dessous permettent une représentation sur l'Apple // de la courbe de Koch pour des niveaux n variant de un à six (limite de définition HGR).

Rappelons ici que la particularité de cette courbe lorsque n = ∞ est d'avoir une longueur infinie et de délimiter une aire finie.

Le type de construction de cette courbe conduit à utiliser un langage autorisant les procédures récursives, ici Pascal.

Le tracé se fera à l'écran (programme FLOCON) ou sur traceur CGP 115 de Radio Shack (programme FLOCTRAC), ce dernier programme étant facile à adapter à tout autre traceur. Pour varier les plaisir, il est possible de remplacer +60 par -60 et -60 par +60 dans la procédure VKOCH.

La méthode :

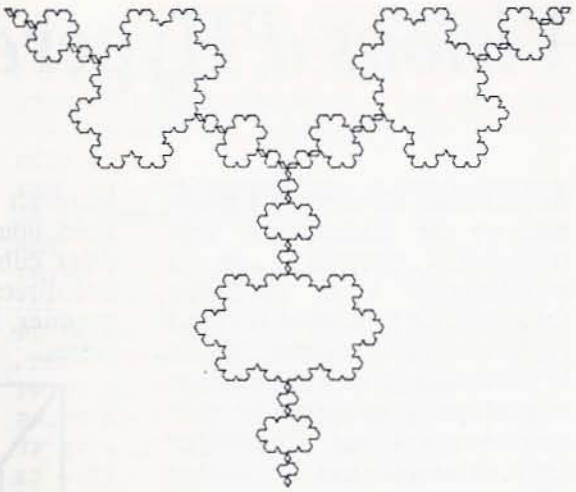
Vous n'avez pas la disquette d'accompagnement : il vous faut saisir et compiler les listings de façon classique.

Vous avez la disquette d'accompagnement : Il faut initialiser une disquette Pascal à l'aide du 'FORMATER', puis il faut lancer le programme 'BASIC-PASCAL' de la disquette Pom's. Il suffit de transférer les fichiers TEXT en suivant les indications données par ledit programme. Il convient alors de compiler et d'exécuter comme habituellement.

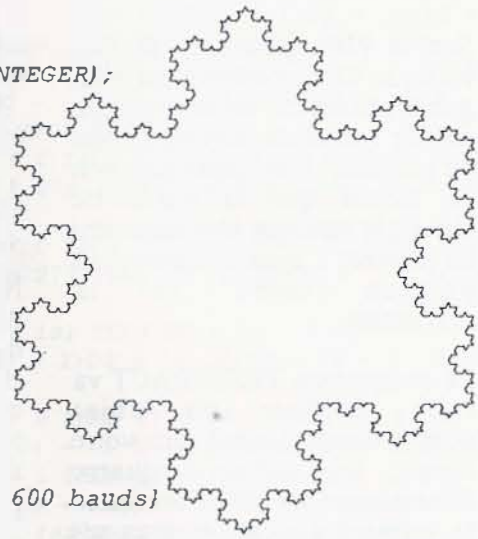
Programme 'FLOCON.TEXT'

```
PROGRAM FLOCON;

  USES TURTLEGRAPHICS,TRANSCEND;

  CONST BASE1=150;

  VAR N,I,ANGLE:INTEGER;
      CAR:CHAR;

  PROCEDURE DEBUT;
   BEGIN
    INITTURTLE;
    PENCOLOR(NONE);
    MOVETO(50,150);
    PENCOLOR(WHITE);
   END;

  PROCEDURE COTE(BASE,DIRECTION:INTEGER);
   BEGIN
    IF DIRECTION>=0 THEN DIRECTION:=DIRECTION MOD 360
                    ELSE DIRECTION:=-((-DIRECTION) MOD 360);
    TURNTO(DIRECTION);MOVE(BASE);
   END;

   PROCEDURE VKOCH(BASE,DIRECTION,NIVEAU:INTEGER);
    BEGIN
     IF NIVEAU=1 THEN COTE(BASE,DIRECTION)
     ELSE
     BEGIN
      VKOCH(BASE DIV 3,DIRECTION,NIVEAU-1);
      VKOCH(BASE DIV 3,DIRECTION+ANGLE,NIVEAU-1);
      VKOCH(BASE DIV 3,DIRECTION-ANGLE,NIVEAU-1);
      VKOCH(BASE DIV 3,DIRECTION,NIVEAU-1);
     END;
    END;

  BEGIN
   REPEAT
     TEXTMODE;
     WRITE('NIVEAU ? ');READLN(N);
     IF N>0 THEN ANGLE:=60
            ELSE BEGIN ANGLE:=-60;N:=-N;END;
     DEBUT;
     FOR I:=0 TO 2 DO VKOCH(BASE1,-120*I,N);
     READ(CAR);
     UNTIL CAR='F';
  END.
```

Programme FLOCTRAC.TEXT



```
PROGRAM FLOCTRAC;

  USES TRANSCEND;

  CONST BASE1=480;
        PI=3.14159265;

  VAR N,I:INTEGER;
      CAR:CHAR;
      LST:INTERACTIVE;


  PROCEDURE COTE(BASE:INTEGER;DIRECTION:REAL);
   BEGIN
    WRITELN(LST,'J',(BASE*COS(DIRECTION)):6:3,',',BASE*SIN(DIRECTION):6:3);
   (trace d'un segment en coordonnees relatives)
   END;

   PROCEDURE VKOCH(BASE:INTEGER;DIRECTION:REAL;NIVEAU:INTEGER);
    BEGIN
     IF NIVEAU=1 THEN COTE(BASE,DIRECTION)
     ELSE
     BEGIN
      VKOCH(BASE DIV 3,DIRECTION,NIVEAU-1);
      VKOCH(BASE DIV 3,DIRECTION+PI/3,NIVEAU-1);
      VKOCH(BASE DIV 3,DIRECTION-PI/3,NIVEAU-1);
      VKOCH(BASE DIV 3,DIRECTION,NIVEAU-1);
     END;
    END;

  BEGIN
   REWRITE(LST,'PRINTER:');
   WRITELN(LST,CHR(9),'7B'); (interface SSC reglee pour 600 bauds)
   WRITELN(LST,CHR(18)); (mise en mode graphique)
   WRITE('NIVEAU ? ');READLN(N);
   WHILE N<>0 DO
    BEGIN
     WRITELN(LST,'M0,',-BASE1 DIV 3); (deplacement absolu)
     WRITELN(LST,'I'); (fixation de l'origine)
     FOR I:=0 TO 2 DO VKOCH(BASE1,-2*PI/3*I,N);
                      WRITELN(LST,'M0,',-BASE1);
     WRITELN(LST,'I');
     WRITE('NIVEAU ? ');READLN(N);
    END;
   WRITELN(LST,CHR(17)); (retour au mode alphanumerique)
   CLOSE(LST);

  END.
```



DOS, ProDOS, Pascal ?

Les disquettes qui accompagnent facultativement Pom's sont enregistrées sous format DOS, car, en effet, il nous faut penser aux nombreuses machines qui ne peuvent utiliser ProDOS ou Pascal.
Certains programmes, par essence, ne sont pas utilisables sous ProDOS (Patches au DOS 3.3 par exemple) et donc ne doivent être convertis.

D'autres ne présenteraient aucun intérêt à être utilisés sous ProDOS (le jeu SNAKE de Pom's 24, par exemple).
Enfin les programmes qui peuvent ou doivent fonctionner sous ProDOS, seront transférés sur une disquette ProDOS. Pour cela, deux solutions se présentent :

- utiliser le programme CONVERT qui est sur l'une des disquettes système ;
- utiliser la disquette 'Utilitaires //c', dans l'option COPIER qui permet de copier des fichiers depuis des disquettes DOS sur des disquettes préalablement formatées ProDOS.

Les fichiers Pascal se trouvent sur la disquette Pom's également en format DOS. Ils seront transférés sur votre disquette Pascal fraîchement formatée, grâce au programme Basic-Pascal de la disquette Pom's.

*Visions d'*Hyper*espace*

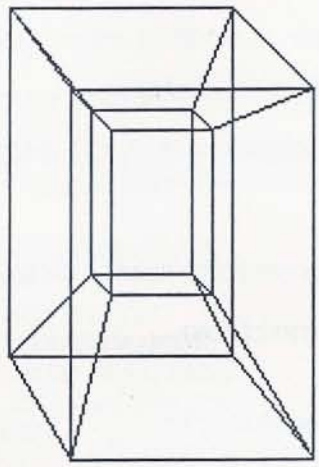
Olivier Sotiriades

Aujourd'hui, les espaces à grand nombre de dimensions sont rencontrés partout : de la psychologie à la physique théorique, de la science-fiction à l'informatique. Pourtant, depuis le raz-de-marée de vulgarisation scientifique lié au centenaire de la naissance d'Albert Einstein en 1979, il semble que le concept d'hyper-espace reste lié, dans l'esprit de chacun, à celle du temps.

Sur le plan purement mathématique, ce rapprochement n'est guère gênant car on ne s'occupe pas de l'unité de mesure (mètre ou seconde). Il est cependant bon de savoir que la Relativité Générale entraîne implicitement un univers à quatre dimensions puisque "tordu" par la gravitation.

Le programme TESSERACT va faire évoluer un objet quadri-dimensionnel sur votre écran, un cube à quatre dimensions : ses arêtes ont toutes la même longueur, et tous ses angles sont droits mais cet objet comporte 16 sommets, 32 arêtes, 24 faces planes et 8 hyperfaces (à 3 dimensions). Petite question : quel est le volume d'un tesseract d'un mètre de côté ?

Pour représenter notre tesseract, il n'est pas inutile d'observer l'analogie avec un cube vu par un être à deux dimensions. Si un tesseract venait à passer devant nous, nous le percevrions comme deux cubes, dont l'un est dans une direction perpendiculaire au premier, reliés entre eux par 8 arêtes.



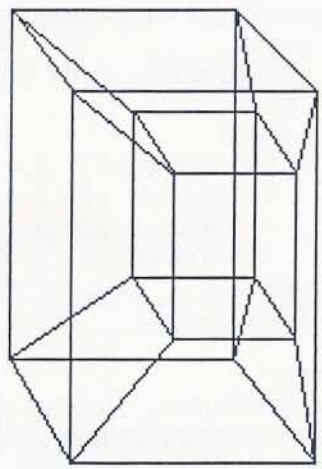
Notre programme, écrit en Applesoft, utilise la particularité du tesseract de disposer d'un chemin hamiltonien. Cela signifie qu'il est possible de parcourir toutes ses arêtes sans avoir à sauter d'un sommet à un autre.

L'algorithme retenu évite de recalculer la position du tesseract après chaque mouvement (car les calculs seraient trop longs), et fournit un affichage le plus aéré possible : 16 des 32 arêtes sont toujours verticales ou horizontales.

On peut faire varier la vitesse de défilement par le paddle 0; l'affichage est gelé par Ctrl-S et Reset permet de sortir du programme.

Enfin, n'oubliez pas que le tesseract a maigri de deux dimensions pour arriver sur votre écran. Vous pouvez toujours en construire un en trois dimensions pour l'observer plus adéquatement. Ou attendre les futures disquettes optiques de Pom's destinées à votre écran holographique...

P.S. : La réponse est un volume de 1 x 1 x 1 x 1 = 1 m^4 bien sûr !



Bibliographie :

Flatland (E.A.Abbott, Présence du futur, Denoël)
La Science et l'Hypothèse (H.Poincaré, Flammarion)
La Quatrième Dimension (R.Rucker, Science Ouverte, Seuil)

Programme 'Tesseract'

```
5  REM*** INITIALISATION ***
10  HGR2 : HGR : POKE  - 16302,0:F = 1: HCOLO
    R= 3:V = .01: PRINT  CHR$ (17): ONERR  G
    OTO 480
20 X1 = 1:X2 = 199:X3 = 80:X6 = 186:Y1 = 1:Y2
     = 136:Y3 = 55:Y6 = 110:K = .3
30 C1 = .5:C2 =  - .5:C3 =  - .5:C6 = .5:D1 =
    .5:D2 =  - .5:D3 =  - .5:D6 = .5:DK = .
    01
35  REM  *** MISE A JOUR DES VARIABLES LIEES
    ***
40 L1 = X2 - X1:X4 = X3 + L1:L2 =  INT (L1 *
    K + .5):X5 = X6 - L2:X7 = X5 +  INT ((X3
     - X1) * K):X8 = X7 + L2
50 M1 = Y2 - Y1:Y4 = Y3 + M1:M2 =  INT (M1 *
    K + .5):Y5 = Y6 - M2:Y7 = Y5 +  INT ((Y3
     - Y1) * K):Y8 = Y7 + M2
60  IF K >  = 1 THEN DK =  - .01
70  IF K <  = .25 THEN DK = .01
80 K = K + DK * F
85  REM  *** TRACAGE ***
90  HPLOT X1,Y3 TO X2,Y3 TO X2,Y4 TO X6,Y8 TO
```

```
X8,Y6 TO X4,Y2 TO X2,Y4 TO X1,Y4
100 HPLOT TO X1,Y3 TO X3,Y1 TO X4,Y1 TO X4,
Y2 TO X3,Y2 TO X3,Y1 TO X7,Y5 TO X8,Y5
110 HPLOT TO X4,Y1 TO X2,Y3 TO X6,Y7 TO X8,
Y5 TO X8,Y6 TO X7,Y6 TO X7,Y5 TO X5,Y7
120 HPLOT TO X6,Y7 TO X6,Y8 TO X5,Y8 TO X7,
Y6 TO X3,Y2 TO X1,Y4 TO X5,Y8 TO X5,Y7 T
O X1,Y3
125 REM *** MISE A JOUR DES VARIABLES LIBRE
S ***
130 IF X1 < = 1 THEN C1 = .5: GOTO 160
140 IF X1 > = 200 THEN C1 = - .5: GOTO 160
150 C1 = C1 * (1 - 2 * (V > RND (1)))
160 X1 = X1 + C1 * F
170 IF X2 < = X1 + 40 THEN C2 = .5: IF X2 <
278 THEN 200
180 IF X2 > = 278 THEN C2 = - .5: GOTO 200
190 C2 = C2 * (1 - 2 * (V > RND (1)))
200 X2 = X2 + C2 * F
210 IF X3 < = 1 THEN C3 = .5: GOTO 240
220 IF X4 > = 277 THEN C3 = - 1: GOTO 240
230 C3 = C3 * (1 - 2 * (V > RND (1)))
240 X3 = X3 + C3 * F
250 IF X5 < = 2 OR X7 < = 2 THEN C6 = 1: G
OTO 280
260 IF X8 > = 276 THEN C6 = - 2: GOTO 280
270 C6 = C6 * (1 - 2 * (V > RND (1)))
280 X6 = X6 + C6 * F
290 IF Y1 < = 1 THEN D1 = .5: GOTO 320
300 IF Y1 > = 130 THEN D1 = - .5: GOTO 320
310 D1 = D1 * (1 - 2 * (V > RND (1)))
320 Y1 = Y1 + D1 * F
330 IF Y2 < = Y1 + 30 THEN D2 = .5: IF Y2 <
190 THEN 360
340 IF Y2 > = 190 THEN D2 = - .5: GOTO 360
350 D2 = D2 * (1 - 2 * (V > RND (1)))
360 Y2 = Y2 + D2 * F
370 IF Y3 < = 1 THEN D3 = .5: GOTO 400
380 IF Y4 > = 189 THEN D3 = - 1: GOTO 400
390 D3 = D3 * (1 - 2 * (V > RND (1)))
400 Y3 = Y3 + D3 * F
410 IF Y5 < = 2 OR Y7 < = 2 THEN D6 = 1: G
OTO 440
420 IF Y8 > = 188 THEN D6 = - 2: GOTO 440
430 D6 = D6 * (1 - 2 * (V > RND (1)))
440 Y6 = Y6 + D6 * F
445 REM *** INTERVERSION DES PAGES HGR ***
450 W = NOT W: POKE - 16299 - W,0: POKE 230
,32 + 32 * W: CALL 62450
455 REM *** PERMET DE FIGER L'IMAGE PAR CT
RL-S ET DE MODIFIER LA VITESSE DE MOUVEM
ENT DU TESSERACT PAR PDL(0) ***
460 HTAB 1: VTAB 1: PRINT " ": IF PDL (0) <
> 255 THEN F = PDL (0) / 51
470 GOTO 40
475 REM *** TRAITEMENT DES (RARES) SORTIES
DE LA FENETRE HGR ***
480 REM
490 PRINT X1;" ";X2;" ";X3;" ";X4;" ";X5;" "
;X6;" ";X7;" ";X8
500 PRINT Y1;" ";Y2;" ";Y3;" ";Y4;" ";Y5;" "
;Y6;" ";Y7;" ";Y8
510 PRINT C1;" ";C2;" ";C3;" ";C6
520 PRINT D1;" ";D2;" ";D3;" ";D6
530 IF X4 > 279 THEN X2 = X2 - 1:C2 = - .5:
X3 = X3 - 1:C3 = - .5:X4 = X4 - 1: GOTO
530
540 IF X5 < 0 THEN X1 = X1 + 1:C1 = .5:X6 =
X6 + 1:C6 = .5:X5 = X5 + 1: GOTO 530
550 IF X7 < 0 THEN X3 = X3 + 1:C3 = .5:X6 =
X6 + 1:C6 = .5:X7 = X7 + 1: GOTO 530
560 IF X8 > 279 THEN X6 = X6 - 1:C6 = - .5:
X2 = X2 - 1:C2 = - .5:X3 = X3 - 1:C3 =
- .5: GOTO 530
570 IF X1 < 0 THEN X1 = 0:C1 = .5: GOTO 530
580 IF X2 > 279 THEN X2 = 279:C2 = - .5: GO
TO 530
590 IF X3 < 0 THEN X3 = 0:C3 = .5: GOTO 530
600 IF X6 > 279 THEN X6 = 279:C6 = - .5: GO
TO 530
610 IF Y4 > 191 THEN Y2 = Y2 - 1:D2 = - .5:
Y3 = Y3 - 1:D3 = - .5:Y4 = Y4 - 1: GOTO
610
620 IF Y5 < 0 THEN Y1 = Y1 + 1:D1 = .5:Y6 =
Y6 + 1:D6 = .5:Y5 = Y5 + 1: GOTO 610
630 IF Y7 < 0 THEN Y3 = Y3 + 1:D3 = .5:Y6 =
Y6 + 1:D6 = .5:Y7 = Y7 + 1: GOTO 610
640 IF Y8 > 191 THEN Y6 = Y6 - 1:D6 = - .5:
Y2 = Y2 - 1:D2 = - .5:Y3 = Y3 - 1:D3 =
- .5: GOTO 610
650 IF Y1 < 0 THEN Y1 = 0:D1 = .5: GOTO 610
660 IF Y2 > 191 THEN Y2 = 191:D2 = - .5: GO
TO 610
670 IF Y3 < 0 THEN Y3 = 0:D3 = .5: GOTO 610
680 IF Y6 > 191 THEN Y6 = 191:D6 = - .5: GO
TO 610
690 RESUME
```

Erratum

Une erreur s'est glissée dans l'article de Pascal Cantot, 'le Kit 65C02' paru dans le Pom's numéro 24. En effet, les instructions TRB, TSB et BIT (adressage immédiat) ne recopient pas les bits 6 et 7 mais, en revanche, positionnent uniquement le bit Z. Plusieurs ouvrages traitant du 65C02 ont commis la même erreur. Merci à Daniel Mueller d'avoir lu avec attention la documentation NCR.

E.P.E. 5.0

Pour lister les programmes Basic en avant et en arrière, pour modifier, insérer, supprimer des caractères sur tout l'écran, sans relire les lignes, pour rechercher toute chaîne de caractères, sous DOS et ProDOS, en 40 et 80 colonnes, sur Apple //e, //e+ et //c.

200,00 F franco
échange 2.0 : 80,00 F

Courrier des Lecteurs

40 pistes sous ProDOS

Intéressé par les articles sur le formatage 40 pistes, j'ai tenté d'adapter les patches sous ProDOS.
Dans le "Beneath Apple ProDOS", des modifications de Filer et de ProDOS sont proposées, mais les essais furent sans succès. Peut-être y a-t-il un bug ? J'ai alors essayé de modifier le source de l'"init ProDOS" du numéro 20 de Pom's mais, à chaque fois j'obtiens un I/O ERROR bien qu'ayant modifié :

- *le nombre de piste ;*
- *le Bit Map ;*
- *le Gap3.*

Quelle est la solution ?

P. Buschini - 75020 Paris

Vous avez raison de signaler une erreur dans le patch proposé par Beneath Apple ProDOS. Dans le courrier des lecteurs du numéro 23 de Pom's, J. Rey indiquait les modifications adéquates. Il indique également celles à faire pour CONVERT ainsi que pour les nouvelles versions du FILER et de ProDOS (version 1.1.1).

Les modifications que vous apportez au programme INIT d'Alexandre Avrane sont correctes, mais il faut également modifier ProDOS. En effet, INIT utilise ProDOS pour vérifier que le disque est bien formaté. Si on ne modifie pas ProDOS, on optient un I/O ERROR quand on veut vérifier un bloc au-delà du n° 280. La modification du Gap3 est par contre inutile, sauf si votre disque est trop rapide.

Les modifications à faire sont les suivantes :

Source INIT. CODE0.S

ligne 318 CMP #40
(au lieu de 35)
ligne 655 : LDY #$27
(au lieu de $22)
ligne 756 : $40
(au lieu de $18)

Objet INIT.CODE0

$223F: 28 (au lieu de 23)
$2491: 27 (au lieu de 22)
$254D: 40 (au lieu de 18)

Objet INIT

$2240: 28 (au lieu de 23)
$2492: 27 (au lieu de 22)
$254E: 40 (au lieu de 12 ou 18)

Notons ici que sur la disquette d'accompagnement, cet octet a la valeur $12, ce qui est incorrect ; la bonne valeur étant $18.

Rappelons les modifications à apporter à ProDOS :

ProDOS 1.0

520D: 40 (au lieu de 18)

ProDOS 1.1.1

56E3: 40 (au lieu de 18)

Attention, même si le formatage se passe bien, certaines pistes peuvent être "recouvertes" par les dernières pistes (le bras du lecteur fait alors un bruit similaire au recalibrage), entrainant des erreurs par la suite. Utilisez l'option "vérifier volume" de FILER ou d'Utilitaire System avant d'utiliser le volume. En cas de problème, réduisez le nombre de pistes.

INIT et CMDLINK

N'y aurait-il pas un bug dans l'article "Allô ProDOS" (Pom's 20) ?
Dans le programme CMDLINK, il me semble qu'il faille remplacer la ligne 30 par : PRINT D$ "BLOAD" C$ ".CODE, A$ 2100". D'autre part, je n'ai pas compris l'utilisation de INIT. Le fichier INIT après intégration par INITLINK et CMDLOAD (après correction) est de type CMD, donc non exécutable. Sur la disquette, il est de type BIN et fonctionne normalement.

L. Fontaine, 72100 Le Mans

Votre correction du programme CMDLINK est parfaitement exacte ; le suffixe est bien ".CODE" et non pas ".LOAD". Vous avez également raison en ce qui concerne le programme INIT obtenu, qui est de type ComManDe, donc pas directement exécutable (il faut le charger puis faire CALL 8192). Sur la disquette d'accompagnement, INIT est de type BINaire ce qui simplifie le problème. Vous pouvez obtenir le même résultat avec CMDLINK en remplaçant, aux lignes 45 et 50, "T$F0" par "TBIN".

EXEC en Pascal

Je n'ai pas réussi à utiliser les fichiers EXEC en Pascal (UCSD 1.2). Le manuel de référence ne comporte pas d'indications sur la commande 'M' du menu principal, et le Pom's numéro 3 ne m'a pas été d'un grand secours.

M. Frank LOPATNIK, 75005 PARIS

Voici la méthode :

- Création du fichier EXEC en mode direct :

```
M        pour Make Exec
TEST     nom du fichier Exec
Y        en réponse à la question :
   Terminator=% Change it ?
%%
F        | Les ordres tapés
L        | ainsi se retrouvent
:        | dans le fichier TEST
RETURN   | tout en étant
Q        | exécutés.
%%       Fin de l'enregistrement
```

L'utilité d'avoir deux '%' est d'éviter de terminer un enregistrement alors que, par

exemple, on est dans l'éditeur et qu'on veut entrer "13%". Bien entendu, un fichier EXEC ne pourra comporter deux '%' consécutifs –ce qui est rare–, sauf à changer alors le 'terminateur'.

- Création ou modification du fichier EXEC par l'éditeur :

  Comme on le voit en éditant TEST, il suffit de commencer par UN '%' et de terminer par QUATRE '%'.

- Exécution au clavier

  Pour exécuter tout fichier, il faut taper X. Ici, au lieu de mettre TEST –ce qu'on aurait fait pour TEST.CODE–, il faut rentrer EXEC/TEST.

- Exécution depuis un programme :

  Pour exécuter tout fichier, il suffit d'employer le SETCHAIN de CHAINSTUFF. Donc :

  `SETCHAIN ('TEST')` pour lancer TEST.CODE et
  `SETCHAIN ('EXEC/TEST')` pour un fichier EXEC.

- Limitation des fichiers EXEC :

  Il n'est pas possible de les 'nester'. Ils ne peuvent contenir de Ctrl-@, Ctrl-A, Ctrl-Z, etc. Si le fichier EXEC lance un programme et que celui-ci fait un READ ou un READLN, il ira chercher dans le fichier EXEC, alors que s'il fait un UNITREAD ou KEYPRESS, il ira chercher au clavier (comme en Applesoft avec GET ou INPUT d'une part, et PEEK(-16384) d'autre part).

Décrypteur

Plusieurs lecteurs nous ont signalé un problème avec l'option 'échange de bits' du Décrypteur (Pom's 21).
Il s'agit en fait plutôt d'un problème d'ergonomie, le programme fonctionnant normalement. Le problème (plantage) survient en fait lorsqu'on lance plusieurs fois de suite l'option échange de bits par un espace au lieu de RETURN. En effet, 'espace' incrémente l'octet 2, ce qui est commode pour les options ET, OU ou EXCL car le filtre est alors pour l'octet 2 : 09 (ORA), 29 (AND) et 49 (EOR).

Dans l'option 'échange de bits', le filtre est 20 (jmp) 6D 10 et incrémenter plusieurs fois l'octet 2 ne peut conduire qu'à des problèmes...

---

et le //c ?

Je possède un Apple //c et il me semble que cette machine n'est pas bien considérée par votre rédaction, certaines allusions le montrent. Bien sûr, les possesseurs de //c ne représentent pas la majorité de votre lectorat mais ils n'ont pas moins envie de connaître leur machine. Voici quelques questions :

- *Comment modifier le 'Hard Copy de la page HGR étendue' pour qu'il fonctionne sur le //c.*
- *Sous ProDOS, si je charge GRAF0 de ToolKit à l'adresse $9500, j'obtiens : No Buffer Avalaible. Comment y remédier, je n'ai pas trouvé la réponse dans ProDOS technical reference manual.*
- *P-FORMAT fonctionne-t-il sur le //c ?*
- *Pour COGO, vous précisez que sur un //c, l'affichage se fait en simple haute résolution, une version //c est-elle prévue ?*

M. Bouland
77300 Fontainebleau

Non, nous ne délaissons pas le //c. Au contraire, nous nous efforçons de modifier les programmes pour qu'ils fonctionnent sur toutes les machines : EPE a ainsi été adapté, le composeur téléphonique, le Basicium. Nous continuons dans cette voie mais il s'agit là d'un problème délicat car les machines ne se laissent pas toujours faire...

- Le programme de Hard Copy fonctionne sans modification : la précision de l'auteur, concernant le SEI n'est à prendre en compte que pour une utilisation simultanée de la souris. Si le programme n'imprime pas, il ne peut s'agir que de la position des 'switches'. Il convient de noter qu'il ne faut pas faire BRUN HARD, mais BLOAD HARD, A$ 4000, puis CALL 16432 (basic) ou 4030G (moniteur).

Plus simplement, vous pouvez utiliser le programme Basic COPIE.

- La réponse se trouve bien dans ProDOS technical reference manual, annexe A.2.1 page 118.
- P-FORMAT fonctionne sur le //c, mais l'éditeur de police qui y est adjoint n'est pas utilisable. Nous étudions la modification.
- COGO fonctionne aujourd'hui normalement sur le //c, c'est le résultat de nos efforts.

Les 64 Ko, le DOS

Comment faire pour utiliser les 64 Ko de la carte 80 colonnes étendue en Basic, et comment faire pour enlever le DOS d'une disquette ?

P. Mykiéta - 92230 Gennevilliers

Le plus simple et le plus pratique est de les utiliser comme disque virtuel : en ProDOS (/RAM), sous DOS 3.3 (Pom's 12, 14 et 16), sous CP/M (Pom's 24). Autrement, il faut utiliser le langage machine et jongler avec les switches et/ou utiliser le sous-programme XFER ($CFCD pour le //c, $C3B0 pour le //e).

Pour enlever le DOS d'une disquette DOS 3.3 il faut, à l'aide d'un éditeur de secteurs, indiquer dans la VTOC que les pistes 1 et 2 sont libres (récupérer la piste 0 est plus délicat). Voici comment faire :

Piste $11, secteur $00, octets $3C, $3D et $40, $41, remplacer 00 00 par FF FF.
En ProDOS, il suffit de supprimer les fichiers ProDOS et BASIC.SYSTEM.

Micro-informations

Jean-Michel Gourévitch

Apple ne s'exclut plus de l'univers IBM et mitonne quelques nouveaux produits de son cru : voici la tendance régnant à Cupertino. Tendance précisée en juin, par Jean-Louis Gassée, le vice-président en charge des nouveaux produits, de passage dans son fief français pour cause d'Apple Expo. Jean-Louis Gassée a donc donné des détails importants sur le positionnement d'Apple, et sur le contenu de la hotte des Pères Noëls de Cupertino. Voici l'occasion de faire le point sur les nouveaux projets. Le présent d'Apple est aux processeurs 16 bits et à la coexistence avec IBM, son avenir lié à celui des communications.

Le nouvel Apple //

Le présent commence pour Apple avec la sortie de l'Apple // version 16 bits. Car, paradoxalement, le récent succès du Macintosh Plus (on recense selon Jean-Louis Gassée une base installée de 750 000 Macs de tous types) pose quelques problèmes. C'est que, parallèlement, le vénérable Apple // qui faisait les choux gras de la firme à la pomme, se vend de moins en moins. Or, si Apple a pu entrer dans les usines, les universités et les commerces, c'était bien grâce au //, transformable au gré de l'utilisateur, en oscilloscope, en machine à apprendre, ou en caisse enregistreuse. Il était donc urgent de le mettre au goût du jour. Ce sera bientôt chose faite. Jean-Louis Gassée a implicitement confirmé toutes les rumeurs sur ce nouvel Apple sauf les dates officielles de sa sortie.

Résumons pour les lecteurs de Pom's : le nouveau // est muni d'un processeur 65C816, tournant à 2,5 Mhz en 16 bits, capable d'adresser 16 Mega-octets de mémoire vive (4 Megas seront directement installables dans la carrosserie du // *new look*). Mais ce processeur peut aussi fonctionner en 8 bits, émuler le 6502 des actuels Apple // et utiliser ainsi tous les logiciels développés pour celui-ci. Physiquement, le dernier né d'Apple se présente comme un vulgaire IBM en trois éléments : unité centrale, clavier détachable et moniteur de visualisation. Parmi ses caractéristiques : une belle résolution graphique (et en 16 couleurs s'il vous plaît), permettant notamment un affichage de 200x320 points ; un nouveau système d'exploitation Prodos intègrant un 'Finder' style Macintosh, et une horloge intégrée. Outre la couleur, l'Apple // pourra aussi disposer d'une carte sonore extrêmement perfectionnée avec synthétiseur à 16 voies. Côtés sorties, cet Apple // modèle 1986 supporte bien entendu le réseau AppleTalk, des lecteurs de disquettes 3 pouces et demi de 800Ko et des lecteurs de 5 pouces 1/4. Bref, voici une machine performante, s'attaquant aux ordinateurs 16 bits modernes, style Atari.

Reste que la clé de sa réussite tient à son prix. Car, dès Noël, le marché de l'informatique individuelle sera durement éprouvé. Des compatibles IBM se négocieront vraisemblablement pour environ 3 000 Francs, mettant la riche bibliothèque de logiciels MS-Dos à la portée des utilisateurs familiaux. Amstrad s'apprête d'ailleurs à sortir un de ces compatibles intégrant notamment le système d'exploitation GEM de Digital Research (au look si 'Macintoshien' qu'on avait dû le modifier sous la pression d'Apple) pour moins de 6 000 Francs. Quant à l'Atari, (vendu 10 000 Francs pour une configuration d'un Mega-octet de mémoire vive), il se verra appuyé par un grand frère plus puissant et devrait encore baisser. Dans ces conditions, Apple devra proposer quelques autres caractéristiques inédites, et jouer serré en matière de tarifs, pour vendre le nouveau // sans mordre sur le Macintosh. En France, Apple a ainsi entamé des manœuvres pour solidifier son réseau (en se séparant de certains concessionnaires, en distinguant ceux appelés à vendre le // de ceux qui peuvent aussi fournir le Macintosh, et en posant à ces derniers des conditions draconiennes, notamment en matière de force de vente, et de surface des locaux).

Aux États-Unis, mêmes procédés complétés par des baisses de prix. En juillet, le prix du disque dur HD 20 a baissé de 300 dollars passant à 1200 dollars (8 400 Francs au cours du change, ça fait rêver, mais le dollar 'informatique' reste plus près de 11 Francs, ce qui met la bête plus près de 13 000 Francs chez nous). Et les lecteurs de 800Ko du Macintosh et de l'Apple // ont baissé de 20% passant à 400 dollars. Bref, comme chez Apple on préfère réserver la nouvelle gamme Macintosh à l'utililisation dans les bureaux, le nouvel Apple // ne sera un vrai succès que si la configuration complète ne dépasse pas ou peu les 10 000 Francs. Pas facile.

Deux gammes distinctes

L'une des révélations de Jean-Louis Gassée, c'est que les gammes resteront séparées. Évaporées, donc les rumeurs qui promettaient le fonctionnement des programmes du Mac sur le nouveau // : «Pas de produits du type couteau suisse» a fermement démenti le vice-président. On verra donc se développer chez Apple dans les deux ans deux gammes bien distinctes comportant chacune trois ou quatre modèles. Ça, c'est une double information : l'Apple // aura au moins un petit frère (en comptant dans la gamme l'ancien // et le //c). Même chose pour le Macintosh, dont la gamme comprendra le récent Mac 512Ko avec lecteur de 800Ko, le Macintosh Plus, le nouveau Mac 'ouvert', et une autre machine. Apple, qui se met à décliner ses gammes comme un constructeur automobile, proposera ainsi vraisemblablement dans chaque ligne un modèle de haut de gamme, un extensible, et un portable. Message reçu cinq sur cinq : attendez vous à voir dans les deux ans un Macintosh portable...

A l'intérieur de chaque gamme, les produits obéiront aux règles de la compatibilité 'ascendante'. En clair : le modèle le plus perfectionné sera capable de lire les programmes de l'ordinateur de base. Sur le papier c'est bien beau, mais quand on a vécu les problèmes de passage du //e au //c, puis du Mac au Macintosh Plus, on demande à voir...

Deux gammes distinctes, donc, mais avec des choses en commun. Et d'abord les périphériques, les lecteurs 800Ko notamment. Bizarrement, c'est en suivant les expériences d'utilisateurs astucieux qu'Apple aurait découvert que les lecteurs Unidisk 3.5 du // pouvaient aussi fonctionner sur le Mac. La leçon a, en tous cas, été retenue. Quant aux disques durs, ils seront aussi communs. Là c'est facile : le nouvel Apple // sera doté comme le Mac et l'Apple //e d'une interface SCSI. Une exception : l'Apple //c, pour lequel l'adaptation d'une interface SCSI aurait été trop diélicate. Ma prévision : l'Apple //c pourrait bien voir son prix baisser considérablement (jusqu'à 3000 Francs) et devenir le bas de gamme d'Apple...

La compatibilité par l'échange des données

Revenons aux choses sérieuses. Si les

deux gammes d'Apple ne pourront s'interchanger les programmes, elles pourront se prêter facilement les données. «Aujourd'hui, a expliqué Jean Louis Gassée, les programmes sont pratiquement 'jetables'. La vraie valeur, ce sont les données saisies, qui coûtent plus cher que le logiciel, et qu'il faut pouvoir récupérer». C'est si vrai qu'actuellement, le problème de la micro-informatique tient dans le coût de ces saisies. Apple propose donc une solution. Puisque les lecteurs et les disquettes sont communs, on devrait donc, avec un utilitaire, pouvoir par exemple récupérer *sur Excel, dans le lecteur d'un Macintosh*, des données saisies *sur AppleWorks et dans le lecteur d'un Apple //*, et vice-versa.

Cette compatibilité des fichiers se réalisant sur le lecteur même du micro-ordinateur et par lecture directe constitue un progrès révolutionnaire. Pourquoi ? Parce qu'elle ne s'arrêtera pas aux frontières d'Apple. C'est qu'IBM vient précisément d'adopter le standard de la disquette 3 pouces et demi pour son nouveau Convertible, et aussi pour ses autres PC. Dans les deux ans, tous les logiciels de l'IBM seront donc portés sur ces disquettes *ainsi que leurs fichiers*. Gassée ne l'a pas caché : on travaille chez Apple sur un utilitaire permettant de lire *dans les lecteurs du Mac et de l'Apple //* les disquettes contenant *les fichiers de l'IBM*, bien que celles-ci soient enregistrées avec un format différent. Voici la première étape de la compatibilité Apple-IBM : celle des fichiers. Il est vraisemblable qu'Apple, qui a déjà eu du mal à reconnaître que la compatibilité des données avec le MS-Dos pouvait apporter un 'plus' à ses clients s'en tiendra là.

La firme de Cupertino laissera à d'autres, comme Dayna ou The Engineering Department, le soin d'implanter sur les ordinateurs à la pomme le système honni MS-Dos. On peut déjà prévoir trois étapes :

- la sortie d'un nouveau 'Mac Charlie' adaptable au Mac Plus, (il se présentera comme un chassis d'extensions ajouté au classique Mac Charlie) permettant de visualiser sur un écran supplémentaire les graphiques aux normes MS-Dos. Ce qui permettrait ainsi de faire tourner le "Flight Simulator" du PC sur le Macintosh (à quoi bon d'ailleurs, puisque la version Macintosh qui vient de sortir en France est sans concurrence ? Mais, il y a tant de sociétés qui exigent du Macintosh la sacro-sainte compatibilité IBM...) ;
- la sortie d'une carte installant dans un slot du nouvel Apple // un ordinateur MS-Dos en réduction (avec coprocesseur 8086, contrôleur de disquettes, etc.) ;
- la même chose pour le nouveau Mac modulaire.

Le pari de la communication

Plutôt que de s'aligner sur les systèmes, on préfère visiblement se réserver chez Apple le pari de profiter du chaos du marché de la communication. Puisque les données sont ce qu'il y a de plus précieux, il faut trouver le moyen de les échanger quel que soit le système et quelle que soit la machine. Connecter des ordinateurs entre eux, ce n'est pas très compliqué. Le problème commence avec les protocoles, l'interface et les structures. Sans parler des normes de transmission différentes en Europe et aux États-Unis.

Voici pourquoi, et c'est une des plus importantes révélations de Gassée, des fonctions de *télécommunications* seront *intégrées* et disponibles *en ressources*, dans les systèmes d'exploitation des Macs et Apple //. Aujourd'hui, lorsqu'un programmeur veut créer une fenêtre dans un programme pour le Mac, il lui suffit d'insérer dans son programme une instruction appelant la routine de création de fenêtres inscrite en ROM. De la même façon, il suffira au programmeur d'un traitement de texte d'appeler les routines de communication pour créer un menu "Accès" permettant, par exemple, sans quitter le traitement de texte, de se connecter sur un serveur ou une base de données, d'ouvrir une fenêtre et d'y obtenir des informations qui pourront ensuite être intégrées par copier/couper/coller. Il s'agira de rendre plus facile et rapide, ce qu'on peut déjà réaliser sur le Mac avec des utilitaires comme Mock Terminal. Et, de la même façon, on pourra extraire des informations d'autres micro-ordinateurs. Ce progrès apportera aux Mac et autres Apple // un certain caractère 'multitâches' : on pourra continuer à travailler sur le traitement de textes, pendant que les informations arriveront. Des perfectionnements qui prennent tout leur sel avec le réseau TOPS, décrit plus bas, permettant à des Mac et des IBM PC de partager fichiers et périphériques.

Pour en avoir plus, il faudra attendre la nouvelle génération de systèmes se profilant après 1989 ou 1990. Avec une architecture du futur, permettant des traitements multiprocesseurs. Ces architectures, développées à l'aide du Cray MXP dont Apple s'est équipé, devront pour voir le jour permettre des améliorations de performances au moins 10 à 20 fois supérieures aux systèmes d'aujourd'hui. Un tel changement jettera alors aux orties la compatibilité. Et bouleversera du même coup la puissance et le paysage de la micro-informatique. Mais on n'en est pas encore là...

Extensions MacPlus : pénurie difficile à résorber

En attendant toutes ces merveilles, il s'agit de résoudre des problèmes bien plus terre à terre. A commencer par la disponibilité des extensions Macintosh Plus. En juin, on avouait chez Apple qu'il faudrait six mois pour résoudre l'embouteillage de demandes. Si ce n'est pas un succès pour le Macintosh Plus... Les derniers servis auront pourtant la satisfaction de se dire que les ROM de leur MacPlus seront moins 'boguées'. En effet, l'un des problèmes du Macintosh Plus tient à des erreurs inscrites dans ses ROM. Ce qui provoque parfois des phénomènes bizarres : un utilisateur ayant acheté deux Macintosh Plus à quatre mois d'intervalles pouvait ainsi faire fonctionner le génial simulateur musical de lecteur de cassettes "Jam Session" sur le plus récent, mais n'obtenait que des bombes sur le plus ancien... Plus sérieusement, ce "bug" empêcherait sur certains Macintosh Plus l'utilisation du port SCSI. Les possesseurs de ces versions là ne s'en sont peut-être pas encore aperçus, car les périphériques SCSI ne sont pas encore légion, mais ça promet encore des tristesses à certains pour demain.

Pour parer à ces ennuis (qui empêchaient aussi certains programmes comme 4e Dimension de tourner), Apple a corrigé ces erreurs par des 'patches' incorporés dans de nouvelles versions du système et du Finder. On en était ainsi en juillet à la version 5.3 du Finder, et à la 3.2c du Système (disquette Pom's 25). Pour la nouvelle imprimante Laserwriter Plus, ce sont des composants fournis par des sous-traitants qui se sont avérés défectueux et ont causé quelques ennuis à ses utilisateurs. Apple France attend sagement la résolution de tous ces problèmes pour l'importer chez nous. Dire que certains stratèges inconscients rêvent encore sans rire de développer les programmes les plus denses et les plus longs jamais réalisés pour les joujoux de la guerre des étoiles version Initiative de Défense Stratégique...

Sur terre, et plus précisément dans la galaxie Apple, on continue à ne pas sombrer dans la monotonie. Avec des extensions, des disques durs et des programmes, comme s'il en pleuvait.

Mémoire : jamais assez

Le Macintosh Plus dispose en standard d'une mémoire vive d'un Mega-octet. Mais, elle peut être facilement quadruplée en remplaçant les barrettes contenant les "puces" de 256Kbits par des puces d'un Mega-bit. C'est un des arguments de vente d'Apple. Quand ? Quand ces super

puces seront disponibles à prix raisonnables. Ce n'est pas le cas, et d'après les spécialistes, il faudra encore patienter, car les puces d'un Mega ne sont encore fournies qu'en petites quantités et à prix importants. Mais, comme les utilisateurs de Macintosh sont des enfants gâtés, ils n'aiment pas patienter. Les constructeurs d'extension l'ont parfaitement compris et se battent pour offrir des extensions maisons... L'impatience n'a pas de prix... Chez **P. Ingéniérie** avec MacMega Plus et pour 9 369 Francs, on dispose de quatre barrettes de 512Ko (réalisées, c'est vrai avec une très belle technologie de montages de composants en surface). On trouve encore des extensions chez **JCR**, baptisées **Megalot** (7 240 Francs pour 2 Megas). Aux États-Unis, on trouve déjà chez **Micro Conversions** des extensions (baptisées **Plus'Plus**) à 4 Megas pour 1400 dollars.

Stockage des données : toujours plus

On vous l'avait dit : depuis que le Mac dispose d'une interface SCSI, les disques durs se bousculent à son connecteur. Mention particulière à ceux proposés par **Pro App.** : on peut les relier aussi bien à la prise lecteur d'un simple Mac, à la prise SCSI d'un Mac plus, qu'à un Apple //. Et, en prime, ils servent de spoolers pour l'impression. Prix modérés : 995 dollars pour 20 megas et 795 pour 10. Les prix baissent : le **Paradise Mac 20** importé par **Micro Conneetion** est ainsi disponible chez nous pour 11 800 Francs. **Alpha Systèmes** importe les disques durs internes **Micah** (25 309 Francs pour 20 Megas) et externes **HD20** (17 671 Francs). Prix bien plus intéressant chez **IEF** qui vend son **Mac Turbo disk** interne de 20 Megas pour 16 485 Francs.

Crex Technology a développé le **Turbo Line**, un disque électronique (il s'agit en fait d'une mémoire vive qui reste en permanence alimentée) particulièrement rapide, car il n'y a pas d'accès disque. Prix : 5 930 Francs pour 1 Mega, 11 480 Francs pour 3 Megas. Pour l'heure, le ruban bleu du disque dur est détenu par l'**AST 4000** et ses 74 Megas. Dommage que cette Rolls du stockage soit si chère : près de 70 000 Francs.

Une sauvegarde pour disque dur d'une capacité de 20 Mega-octets sur cassette numérique, le **Storm 20** est proposée chez **DIF électronic** pour 19 600 Francs.

Réseaux : du nouveau

Du disque dur au serveur de réseau, il n'y a parfois que l'épaisseur d'un logiciel. C'est le cas avec **Asmodée d'ACI**. Cet utilitaire, qui s'installe dans le menu ⌘ du Macintosh permet, à distance, et sous AppleTalk d'accéder aux fichiers stockés sur la disquette ou le disque dur d'un autre Macintosh.

La grande nouveauté, en matière de réseaux, c'est **TOPS**, compatible avec AppleTalk, encore distribué par **Micro Connection**. La réponse à bien des problèmes d'incompatibilité, ce réseau local permet de faire partager des fichiers et des périphériques par des Macs, mais aussi des IBM PC ou compatibles. Pour un prix très raisonnable, car le coût de connection est de 2 080 Francs par Macintosh et 5 634 par PC ou compatible (il faut une carte). On n'a pas fini d'entendre parler de TOPS.

Unix

L'une des nouveautés annoncées par Jean-Louis Gassée, c'est l'adoption de systèmes sous **Unix** qui sera disponible dès 1987 pour le Macintosh. Pour cela, Apple a acheté la technologie développée par **Cadmus**. L'adoption d'Unix permettra d'implanter le futur Mac haute de gamme modulaire sur le marché de la station de travail scientifique. A remarquer un indice : Apple propose depuis peu aux États-Unis **MacWorkStation**. Un programme permettant de se relier à une unité centrale en éliminant l'émulation jusque là obligatoire des terminaux DEC VT 100 ou IBM 3278, mais en conservant la facilité du Mac de ses menus et de ses fenêtres. Ce programme destiné à des non techniciens d'accéder aux programmes de l'unité centrale. Il comprend le source, le logiciel MacHost (source pour système d'exploitation Unix V et VMS) et une documentation. Sa licence d'utilisation dans un programme revient à 1500 dollars.

Gestion de bases de données

Controle X, qui a divorcé d'avec **Version Soft**, sort enfin **CX 500**, l'évolution relationnelle, multifichier et paramétrable de MacBase. J'attendrai d'avoir vu tourner la version définitive pour en dire plus. **Soulat Frères** importent **Overvue 2**. Cette version est particulièrement rapide et simple. Prix : 3 320 Francs. **Blyth Software** a sorti une nouvelle version d'**Omnis 3** c'est **Omnis 3 Plus**. A noter que ce programme s'est tranquillement installé comme la meilleure base de données pour Mac (et aussi la plus sûre et la moins 'boguée'). Dans la nouvelle version, des champs pour des heures avec possibilité de calculs sur heure, des champs de texte pouvant comporter plusieurs lignes avec passage à la ligne automatique, la possibilité de créer des bandes de défilement pour passer rapidement à la fin des données, 60 nouvelles macros, la possibilité d'avoir des fichiers de 160 Megas, la possibilité de lancer depuis Omnis d'autres programmes pour le Mac, des sauvegardes automatiques, etc. Omnis est importé en France par **KA informatique**. A remarquer enfin, l'amélioration de **Filevision**, l'un des premiers programmes pour le Mac, avec **Business Filevision**. Cette dernière version permet notamment l'importation de dessins de MacPaint, MacDraw, MacDraft, ou d'images numérisées, ainsi que le traitement graphique de données crées dans File, Omnis 3, Jazz, Excel, etc.

Revoilà les intégrés

Programmes intégrés, ou programmes reliés par switcher? On se croirait revenu un an en arrière, au moment où la querelle faisait rage entre Lotus, qui avait sorti Jazz –son programme intégrant plusieurs applications– et Microsoft, sortant Excel, un programme intégrable. Les revoici tous deux en piste.

Lotus sort la nouvelle version 1A de **Jazz** pour le Macintosh Plus. Vitesse accrue, capacité de traitement plus importante, sur disquette 800Ko. En prime, une sérieuse baisse : Jazz, ne coûte plus que 3 500 Francs.

Microsoft, qui s'était affiché comme l'ennemi des intégrés, met la dernière main à **MacWorks**. La revue britannique **Practical Computing** a pu mettre la main sur une version de test et la... tester. Conclusions : une base de données semblable à File, mais sans possibilité d'y incorporer des graphiques, un programme de communications simple, un tableur puissant (9999 rangées sur 255 colonnes) mais limité, avec 50 fonctions, et la possibilité de convertir les chiffres en graphiques (mais avec peu de types de graphiques), et surtout un très bon traitement de texte avec notamment la possibilité d'y dessiner des lignes, des cercles, des carrés. On peut aussi y concevoir des documents plus larges que l'écran, qui seront imprimés en largeur, récupérer des documents MacWrite, etc. MacWorks devrait être vendu moins de 300 dollars aux États-Unis.

Des traitements de texte en pagaille

La décision d'Apple de ne plus livrer systématiquement MacWrite avec le Mac continue à susciter la création de traitement de texte. **Microsoft** se prépare aussi à sortir une nouvelle version de **Word**, avec correcteur d'orthographe incorporé, traitement

d'idées style Think Tank, lui aussi incorporé, génération automatique d'index et de tables des matières, etc. En France, **ACI** lancera à la rentrée **Writer Plus**. Un traitement de texte multi-fenêtres, permettant de réaliser des mailings, avec plusieurs colonnes, césure et indexation automatiques. **Mac Author**, le traitement de texte avec feuilles de style, possibilité d'amalgamer les graphiques au texte, espacement réglable, etc. dont j'avais parlé dans cette rubrique est maintenant importé chez nous par **Italsoft**. Prix : 2 312 Francs.

Aux États-Unis, on trouve chez **Icon Review**, le traitement de texte **MultiWrite**, un MacWrite multi-fenêtres intégrant un traitement d'idées, un compteur de caractères, etc. pour 80 dollars.

Nouveauté d'importance, **Think Tank** de **Living Videotext** a désormais un successeur. C'est **More**. Il offre en plus la possibilité d'illustrer graphiquement le cheminement des idées par des 'arbres' Le programme comprend aussi un calculateur intégré additionnant les nombres entrés. Si ce programme, qui sera vendu 300 dollars, est aussi révolutionnaire que l'avait été, en son temps, Think Tank, ça promet.

Éditique

C'est le nouveau nom, pas très beau, pour traduire ce que les Américains appellent *Desktop publishing*, et qui est en fait de l'édition électronique. C'est toujours la grande mode pour le Macintosh. **Concept Publishing Systems** a sorti **Adnet**. Le système complet comprend un Macintosh équipé d'un disque dur interne de 10 Megas, et spécialement configuré, une tablette graphique et une LaserWriter. Prix : 17000 dollars.

Plus abordable, **Glue** de **Solutions Inc.** offre pour 49 dollars la possibilité de transférer facilement dans **PageMaker** ou **Ready Set Go** des graphiques créés dans **Excel** ou **Jazz**. Ces images peuvent être rappelées et transférées dans le programme d'édition par couper/coller. Précisément, une nouvelle version -la 3.0- de **Ready Set Go** est en préparation avec possibilité de travailler sur plusieurs documents, nombre de pages accru, et automatisation de certaines fonctions de conception de documents.

Un drink ?

On a pu apprécier, dans la chaleur d'Apple Expo, **Mac Drink** sur le stand de **Micro Valley**. Le logiciel **AutoMac** de **Pro Forma** pilotant un Mac relié à des bouteilles permettait de doser des cocktails. L'ensemble est sérieux. Il se compose d'un Mac, d'un Apple II sans lecteurs, et de cartes interface. Le logiciel est écrit en Prolog IA. Il permet de commander des expériences de chimie, électricité, robotique et de piloter un système à électro vannes, un bras articulé, ou... un train électrique. Une telle débauche technologique pour un train électrique, c'est un peu de l'overkill, non ?

Côté Apple //

Alpha Systèmes importe une série de nouveautés pour l'Apple //. Parmi elles : **Kyan**, un Pascal sous Prodos avec facilités d'utilisation comparables à Unix, routines graphiques, etc. pour 1 234 Francs ; **Unimate**, un logiciel pour lecteur de 800Ko permettant d'utiliser ce lecteur avec le DOS 3.3, le Pascal et le CPM pour 580 Francs ; et la carte **Checkmate** avec processeur 16 bits 65C816 pour 3 150 Francs.

Apple distribue **Extasie**, un logiciel de dessin permettant de mélanger dans une même image les 16 couleurs disponibles et la très haute résolution graphique noir et blanc de 560 points par ligne. On peut y construire des rectangles et ellipses, on dispose d'une loupe, d'un aérographe, les sorties peuvent s'effectuer en couleurs sur une imprimante Imagewriter II. Prix : 699 Francs.

Citizen, distribué par **Geveke**, a conçu une interface permettant de relier une de ses imprimantes (souvent bon marché) à un Apple //. Prix : 1 000 Francs.

Beagle Bros, un des pionniers de la programmation pour Apple //, a développé **Macro Works**, un ensemble d'améliorations pour Appleworks comprenant notamment des macros, la possibilité d'impression en colonne, etc. Prix : 35 dollars.

Et des traitements de texte

Multi Scribe de **Styleware** est un traitement de texte utilisant la double haute résolution graphique, très semblable à MacWrite pour le Mac. Comprenant même un éditeur de caractères. Prix 60 dollars.

L'un des ancêtres des traitements de texte pour le //, **Format 80** d'**Elite Software** peut maintenant incorporer des formules mathématiques. Grâce à **Format 80 Scientific**.

Toujours IBM

Cette fois, c'est le //c qui peut lire et écrire des fichiers compatibles IBM grâce à une carte baptisée **Envoy**, et conçue par **Asky** aux États-Unis. La carte peut contrôler les lecteurs 5 pouces 1/4 et 3 pouces 1/2 Apple, mais aussi des lecteurs IBM, y compris celui de 1,2 Megas de l'AT. En reliant un lecteur compatible IBM à l'Apple, et grâce à un logiciel de conversion, on obtient des fichiers pouvant être relus sur un IBM. Le prix de la carte n'est que de 180 dollars, avec le logiciel, mais sans le lecteur IBM. Selon ses promoteurs, le système permet ainsi à des employés d'une société d'emporter chez eux des disquettes de données IBM et de les relire et les retravailler chez eux sur leur Apple. Alleluia!

Rectification

Dans le dernier numéro de Pom's, il fallait bien évidemment lire **Pixel Studio**, à la place de **Mac Palette**. Pixel Studio, c'est bien le nom de l'ensemble graphique couleur pour le Mac.

Adresses

Alpha Systèmes - 29, Bd Gambetta
38000 Grenoble - Tél. : 76 43 19 97
P.Ingénierie - 226, Bd Raspail
75006 Paris - Tél.: 43 21 93 36
JCR - 58, rue Notre Dame de Lorette
75009 Paris - Tél. : 42 82 19 80
Pro App - 1475 S. Bascom Avenue
Campbell CA 95008
Micro Connection
103/105, rue du Château
92100 Boulogne - Tél. : 48 25 83 83
IEF - 217, quai Stalingrad
92130 Issy les Moulineaux
Tél. : 45 57 14 14
Crex Technology - 34, rue Poncelet
75017 Paris - Tél. : 42 67 80 46
DIF Electronic
94, Bd du Montparnasse
75014 Paris - Tél. : 43 21 43 65
ACI - 6, avenue Franklin Roosevelt
75008 Paris - Tél. : 43 59 89 55
Soulat Frères
45/47, rue de la division Leclerc
94250 Gentilly - Tél. : 47 40 00 20
Italsoft - 14, rue Vauvenargues
75018 Paris
Icon Review
PO Box 2566 Monterey
CA 93942
Concept Publishing System
126 Monroe St Beaver Dam - WI 53916
Pro-Forma - 14, rue Martel
75010 Paris - Tél. : 45 23 25 05
Geveke Electronique
2 à 18 rue des Peupliers
92000 Nanterre - Tél. : 47 80 96 96
Beagle Bros
3990 Old Town Avenue
San Diego CA 92110
Styleware - 6405 Hillcroft Suite
201 Houston Texas 77081
Elite Software
4 Hawthylands Drive, Hailsham
East Sussex BN27 1HE, UK

Bon de commande

Disquettes

HAIFA source	(cf. Pom's n° 5)	.....	à 60,00 F	.....
H-BASIC	(cf. Pom's n° 8)	.....	à 150,00 F	.....
MUSIC	(cf. Pom's n° 10)	.....	à 80,00 F	.....
DISK-MANAGER	(cf. Pom's n° 11)	.....	à 450,00 F	.....
BASICIUM	(cf. Pom's n° 13)	.....	à 150,00 F	.....
E.P.E. 5.0	(cf. Pom's n° 23)	.....	à 200,00 F	.....
Échange E.P.E. 5.0	(cf. Pom's n° 23)	.....	à 80,00 F	.....
PASCAL	(cf. Pom's n° 15)	.....	à 80,00 F	.....
MAX (Moniteur étendu)	(cf. Pom's n° 18)	.....	à 150,00 F	.....
DOMINOS	(cf. Pom's n° 19)	.....	à 80,00 F	.....
P-FORMAT][, P-POLICE][	(cf. Pom's n° 21)	.....	à 200,00 F	.....
COGO	(cf. Pom's n° 21)	.....	à 150,00 F	.....
LUDOLOGIC	(cf. page 8)	.....	à 80,00 F	.....

Recueils

N°1, recueil des revues 1 à 4	.....	à 140,00 F	.....
Disquettes d'accompagnement 1 à 4	.....	à 200,00 F	.....
N°2, recueil des revues 5 à 8	.....	à 140,00 F	.....
Disquettes d'accompagnement 5 à 8	.....	à 200,00 F	.....
N°3, recueil des revues 9 à 12	.....	à 140,00 F	.....
Disquettes d'accompagnement 9 à 12	.....	à 200,00 F	.....

Revues, disquettes

Revues 4 7 8 *(n°9 épuisé)*	.....	à 35,00 F	.....
Revues 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25	.....	à 40,00 F	.....
Disquettes Apple II, //e, //c 1/2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25	.....	à 60,00 F	.....
Disquettes Macintosh			
14/15/16 groupées	.....	à 150,00 F	.....
17 18 19 20 21 22 23 24 25	.....	à 80,00 F	.....
Mac 'A'	.....	à 80,00 F	.....
MacAstuces	.....	à 200,00 F	.....
"Raccourci"	.....	à 200,00 F	.....

Abonnements

Pour 6 numéros à partir du n° ☐

Abonnement à la revue seule	.....	à 200,00 F	.....
Abonnement revue + disquettes Apple II, //e, //c	.....	à 500,00 F	.....
Abonnement revue + disquettes Macintosh	.....	à 600,00 F	.....

Total TTC :

Supplément avion hors CEE : 15,00F par numéro et/ou disquette : _____

Montant du règlement : _____

Envoyez ce bon et votre règlement à : EDITIONS MEV, 64 rue des Chantiers 78000 VERSAILLES

Nom : .....

Adresse : .....

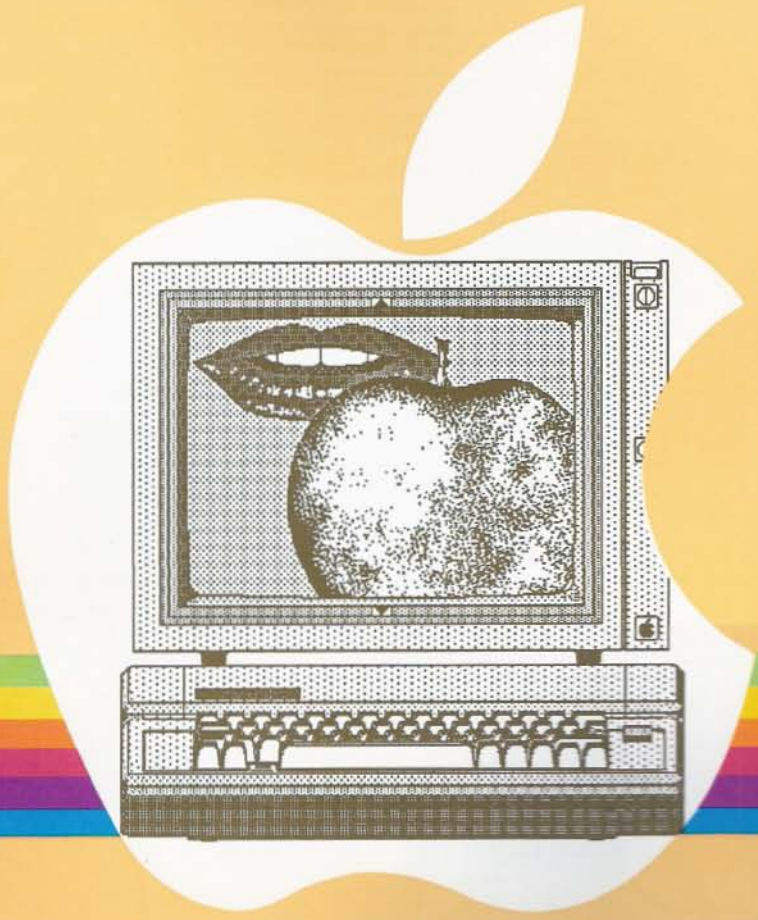
Imprimerie Rosay, 94300 Vincennes, Imprimé en France. Dépôt légal : juillet 1986. N° 3449

pom's

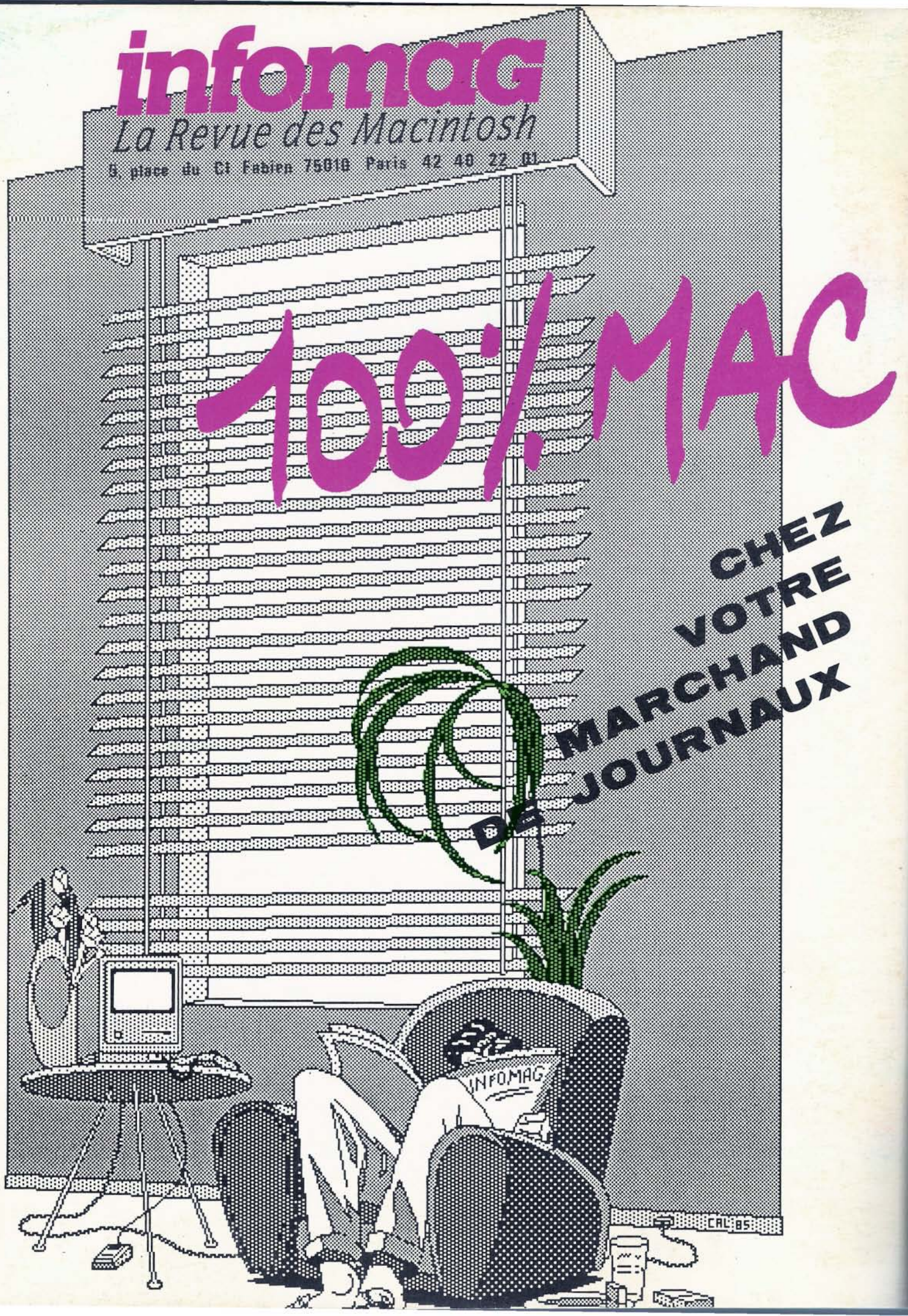
Recueil n° 3

Programmes, Trucs, Astuces, Initiation

Apple][+, //e, //e+, //c



Basic, Pascal, Assembleur...


infomag
La Revue des Macintosh
5, place du Cl Fabien 75010 Paris 42 40 22 01
100% MAC
CHEZ VOTRE MARCHAND DE JOURNAUX
INFOMAG
CAL 85